

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Государственное образовательное учреждение

высшего профессионального образования

КЫРГЫЗСКО-РОССИЙСКИЙ СЛАВЯНСКИЙ УНИВЕРСИТЕТ

имени Первого Президента Российской Федерации Б.Н. Ельцина

ЕСТЕСТВЕННО – ТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра информационных и вычислительных технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему:

**Разработка программных средств для автоматизации бизнес-процессов компании по
ремонту квартир**


Выполнил студент группы ЕПИ-2-19

Касымжапаров Адилет



Руководитель

к.т.н., доцент Хмелева И.В.



Работа к защите допущена

заведующей кафедрой ИВТ

д.т.н., проф. Лыченко Н.М.



БИШКЕК 2023

Аннотация

Тема дипломной работы – «Разработка программных средств для автоматизации бизнес-процессов компании по ремонту квартир».

Цель работы – Разработать программную систему закрывающую потребности компании в автоматизации таких бизнес-процессов как, контроль качества, отслеживание заказа, обработки заказа и др., а также обеспечить хранение данных.

Данное программное средство востребовано по нескольким причинам: улучшение коммуникации и взаимодействия между сотрудниками компании, а также клиентами; управление задачами и ресурсами компании;

Для реализации были использованы следующие средства разработки: PyCharm, Visual Studio Code, pgAdmin 4.

Языки программирования: Python, JavaScript, TypeScript.

Доступ к данным: ORM SQLAlchemy.

База данных: PostgreSQL.

Контроль версий: Git.

Библиотеки: React, Vite.

Разработанная система включает в себя следующие компоненты:

- Web-приложение для пользователей.
- Web API.
- База данных для хранения.

Объем пояснительной записки: 103 страницы

Количество рисунков: 48

Количество таблиц: 7

Annotation

The topic of the thesis - "Development of software to automate business processes of the company to repair apartments.

Purpose of work - To develop a software system that closes the needs of the company to automate such business processes as quality control, order tracking, order processing, etc., as well as to provide data storage.

This software tool is in demand for several reasons: to improve communication and interaction between company employees, as well as customers; to manage the tasks and resources of the company;

The following development tools were used for the implementation: PyCharm, Visual Studio Code, pgAdmin 4.

Programming languages: Python, JavaScript, TypeScript.

Data access: ORM SQLAlchemy.

Database: PostgreSQL.

Version control: Git.

Libraries: React, Vite.

The developed system includes the following components:

- Web-application for users.
- Web API.
- A database for data storage.

Volume of the explanatory note: 103 pages

Number of figures: 48

Number of tables: 7

Аннотация

Дипломдук иштин темасы - "Батирлерди оңдоо боюнча компаниянын бизнес-жараяндарды автоматташтыруу үчүн программалык камсыздоону иштеп чыгуу".

Иштин максаты-сапатты контролдоо, заказды көзөмөлдөө, заказды иштеп чыгуу ж.б. сыяктуу бизнес-процесстерди автоматташтырууда компаниянын жабуучу керектөөлөрүн программалык камсыздоо системасын иштеп чыгуу, ошондой эле маалыматтарды сактоону камсыз кылуу.

Ишке ашыруу үчүн иштеп чыгуунун төмөнкү каражаттары колдонулду: PyCharm, Visual Studio Code, pgAdmin 4.

Программалоо тилдери: Python, JavaScript, TypeScript.

Маалыматтарга жетүү: ORM SQLAlchemy.

Маалымат базасы: PostgreSQL.

Версияны башкаруу: Git.

Китепканалар: React, Vite.

Бул программа бир нече себептерден улам талап кылынат: компаниянын кызматкерлеринин, ошондой эле кардарлардын ортосундагы байланышты жана өз ара аракеттенүүнү жакшыртуу; компаниянын милдеттерин жана ресурстарын башкаруу; Иштелип чыккан система төмөнкү компоненттерди камтыйт:

-Об-колдонуучулар үчүн колдонмо.

- Web API.

- Сактоо үчүн маалымат базасы.

Түшүндүрмө каттын көлөмү: 103 барак

Чиймелердин саны: 48

Таблицаалардын саны: 7

Оглавление

Введение.....	5
Глава 1. Видение программного продукта.....	7
1.1. Введение.....	7
1.2. Позиционирование программного продукта	8
1.3. Описание пользователей	9
1.4. Краткий обзор программного продукта.....	10
1.5. Возможности продукта.....	11
1.6. Ограничения.....	12
1.7. Показатели качества	12
1.8. Старшинство и приоритеты.....	13
1.9. Другие требования к программному продукту.....	13
1.10. Требования к документации.....	14
1.11. Маркировка и пакетирование.....	15
Глава 2. Спецификация требований к программному продукту.....	16
2.1. Введение.....	16
2.2. Общее описание	18
2.3. Специфические требования.....	20
Глава 3. Проектирование и конструирование ПО	31
3.1. Введение.....	31
3.2. Разработка диаграммы классов	31
3.3. Разработка диаграммы компонентов	34
3.4. Разработка диаграмм последовательностей.....	35
Глава 4. Разработка тестов и тестирование ПО	43
4.1. Введение.....	43
4.2. Атрибуты качества.....	43
4.3. Подход к тестированию	50
4.5. Расписание тестирования	52
4.6. Критерии завершения	54
4.7. Ограничения и исключения.....	54
Глава 5. Руководство пользователя.....	56
5.1. Назначение системы	56
5.2. Условия применения системы.....	56
5.3. Подготовка системы к работе.....	58
5.4. Описание операций.....	59

5.5. Аварийные ситуации	83
Заключение.....	84
Список использованных источников	85
Приложение 1. Глоссарий	87
Приложение 2. Листинг	88

Введение

Строительные компании сталкиваются с рядом проблем, таких как сложность управления заказами, контроль исполнения работ, координация взаимодействия с клиентами и подрядчиками, а также необходимость генерации отчетности и анализа данных. Для решения этих проблем требуется внедрение современных программных средств, способных автоматизировать и оптимизировать бизнес-процессы компаний по ремонту квартир.

Объектом исследования данной работы являются бизнес-процессы компании по ремонту квартир, такие как: оформление заказов, контроль исполнения работ, взаимодействие с клиентами и подрядчиками, а также генерация отчетности и анализ данных. Предметом исследования являются программные средства, разработка которых позволит автоматизировать указанные бизнес-процессы и повысить эффективность работы компании.

Современное состояние проблемы исследовано в ряде источников, указывающих на актуальность автоматизации бизнес-процессов в компаниях, занимающихся ремонтными услугами. Однако, системы-аналоги, предлагаемые на рынке, не всегда полностью удовлетворяют требованиям компаний, так как не всегда отвечают специфическим требованиям бизнес-процессов компаний по ремонту квартир. Поэтому требуется разработка собственных программных средств, соответствующих конкретным потребностям компании.

Целью данного исследования является разработка программных средств, предназначенных для автоматизации бизнес-процессов компании по ремонту квартир, с использованием современных методов и подходов объектно-ориентированного анализа и проектирования на языке UML. Для достижения данной цели были поставлены следующие задачи: проведение анализа существующих бизнес-процессов компании по ремонту квартир, выявление их основных проблем и узких мест; изучение существующих систем-аналогов и их сравнительный анализ; разработка требований к программному обеспечению, учитывающих специфику бизнес-процессов компании по ремонту квартир; проектирование архитектуры программного решения на основе методов и подходов объектно-ориентированного анализа и проектирования; разработка программных средств на выбранной платформе, включая создание пользовательского интерфейса, реализацию функциональности, тестирование и оптимизацию системы; внедрение разработанного программного решения в компанию по ремонту квартир и его апробация на практике; анализ эффективности внедренного программного решения на основе сравнения

показателей до и после внедрения, оценка экономической эффективности использования разработанного ПО.

Структура программных средств включает несколько модулей, таких как модуль управления заказами, модуль контроля исполнения работ, модуль взаимодействия с клиентами и подрядчиками. Они взаимодействуют между собой и обеспечивают автоматизацию основных бизнес-процессов компании по ремонту квартир, позволяя оптимизировать деятельность компании, упростить процессы работы с заказами, повысить контроль исполнения работ, облегчить взаимодействие с клиентами и подрядчиками, а также улучшить отчетность и анализ данных.

Результаты данной работы могут быть использованы компаниями, занимающимися ремонтными услугами, для автоматизации своих бизнес-процессов и повышения эффективности работы. Данное исследование внесет вклад в развитие области автоматизации бизнес-процессов и разработки программных средств для ремонтных компаний.

Документ состоит из 5 основных частей: Видение, Спецификация требований, Проектирование ПО, Разработка тестов и Руководство пользователя

1. Видение описывает возможности продукта, его ограничения, а также краткий обзор продукта, который поможет понять его функционал.
2. Спецификация требований показывает возможности системы, а также требования, заложенные для разработки ПО.
3. Проектирование ПО описывает архитектуру продукта, описание структуры базы данных, структуру кода, а также то, как отдельные процессы взаимодействуют друг с другом.
4. Разработка тестов - глава, которая показывает работоспособность системы с помощью модульного и интеграционного тестирования.
5. Руководство пользователя – глава, где разъясняется то, как пользоваться системой, выполнять то или иное действие с приложенными скриншотами рабочей системы.

Глава 1. Видение программного продукта

1.1. Введение

1.1.1. Цель

Документ описывает характеристики программного продукта для автоматизации бизнес-процессов компании по ремонту квартир, а также формирует четкие и общие понимания целей, направления развития и потенциальной ценности продукта для клиентов, пользователей и бизнеса в целом.

Документ описывает каким образом программный продукт (ПП) будет решать проблемы и удовлетворять потребности целевой аудитории, какие новые возможности он предоставит, какие рыночные тренды учитываются, а также определяются цели, миссия и стратегия развития продукта.

Документ предназначен для лиц, разрабатывающих, тестирующих и управляющих данным продуктом для общего понимания того, куда они движутся и какие результаты должны быть достигнуты, чтобы продукт успешно конкурировал на рынке.

1.1.2. Контекст

Описание ПП включает в себя сферу деятельности компании по ремонту квартир, а также смежные области, такие как контроль качества, логистика и т.д. В данном контексте ПП будет использоваться для автоматизации бизнес-процессов, связанных с выполнением ремонтных работ в квартирах и обслуживанием клиентов.

ПП должен обеспечивать удобный интерфейс для работы с клиентской базой, возможность прикрепления договоров на ремонтные работы и учета затрат на материалы и работу. Он также должен позволять отслеживать текущий статус выполнения работ, распределять задачи между исполнителями и контролировать сроки выполнения заказов.

В общем, ПП должен решать широкий спектр задач, связанных с бизнес-процессами компании, и помогать оптимизировать их выполнение в целях повышения эффективности и удовлетворенности клиентов.

1.1.3. Определения, акронимы и сокращения

Определения приведены в документе «Приложение1. Глоссарий проекта».

1.1.4. Ссылки

При создании документа использовались следующие ссылки, вынесенные в раздел «Список используемой литературы»: [1] – [8].

1.1.5. Краткое содержание

В данной главе будет определен контекст, описываемой системы, а также детализируемые требования к программному продукту, позиционирование продукта на рынке и целевую аудиторию, которую он предназначен обслуживать. Также будет рассмотрены функциональные возможности продукта, его ограничения и показатели качества, а также установлены требования к документации и упаковке. В результате, читатель получит общее представление о программном продукте (ПП) и его важных характеристиках, а также поймет какие потребности он может удовлетворять и каким образом можно улучшить бизнес-процессы компании.

1.2. Позиционирование программного продукта

1.2.1. Деловые преимущества

ПП, предназначенный для автоматизации бизнес-процессов компании, предоставляет следующие деловые преимущества:

- Увеличение производительности бизнес-процессов: автоматизация рутинных операций, таких как прикрепление договоров, учет затрат на материал и работу, позволяет сократить время на выполнение этих задач и повысить производительность компании в целом.
- Улучшение качества обслуживания клиентов: ПП обеспечивает удобный интерфейс для работы с клиентской базой, что позволяет быстро и эффективно обрабатывать запросы клиентов, составлять договоры на ремонтные работы и проводить учет затрат.
- Снижение затрат на обслуживание клиентов: ПП позволяет сократить количество ошибок и снизить затраты на обслуживание клиентов.
- Улучшение контроля над бизнес-процессами: ПП предоставляет возможность контроля качества ремонтных работ и учета затрат на материалы и работу, что позволяет компании более эффективно управлять своей деятельностью и принимать правильные решения.
- Увеличение конкурентоспособности компании: использование современных технологий и автоматизация бизнес-процессов позволяет компании опережать конкурентов и предоставлять клиентам более высокое качество услуг [1].

1.2.2. Определения проблемы

Одной из главных проблем является необходимость ведения учета затрат на материалы и работу при выполнении ремонтных работ. Ручной учет может быть

неэффективным, ошибочным и время-затратным, что приводит к недостаточной точности расчетов и увеличению затрат на материалы и работу.

Кроме того, существует проблема управления клиентской базой и составления договоров на ремонтные работы. Ручное ведение базы клиентов может быть неудобным и ненадежным, а составление договоров может занимать много времени и потребовать дополнительных ресурсов.

Также компания сталкивается с проблемой контроля качества выполнения ремонтных работ. Ручное контролирование может быть неэффективным и приводить к недостаточной точности оценки качества работ.

ПП для автоматизации бизнес-процессов компании по ремонту квартир предназначен для решения этих проблем и повышения эффективности работы компании.

1.2.3. Определение позиции программного продукта

ПП позиционируется как инновационное решение для управления бизнес-процессами и повышения эффективности компании. Он обладает уникальными функциональными возможностями, которые позволяют автоматизировать процессы, ускорить выполнение работ и улучшить качество обслуживания клиентов. Продукт будет предложен как решение для средних и крупных компаний, занимающихся ремонтом квартир, которые стремятся оптимизировать свои бизнес-процессы и улучшить качество обслуживания своих клиентов. Кроме того, продукт будет представлен как инструмент для управления проектами, что позволит клиентам быстро и эффективно управлять работами и контролировать бюджет проекта.

1.3. Описание пользователей

1.3.1. Сведения о пользователях

- Менеджер – сотрудник компании, отвечающий за обработку заказа и назначение ответственных за заказ.
- Клиент – пользователь, оставляющий заявку на ремонт.
- Замерщик – сотрудник компании, занимающийся замером объекта ремонта и внесением данных по замеру.
- Дизайнер - сотрудник компании, занимающийся созданием дизайна, визуализацией объекта.
- Строитель - сотрудник компании, занимающийся предоставлением фотоотчета по состоянию ремонта.

1.3.2. Пользовательская среда

Пользовательская среда ПП разрабатывается с учетом особенностей работы каждой категории пользователей.

Менеджеру доступен интерфейс для управления заказами, контроля за сроками выполнения работ и созданию сотрудников. Клиентам предоставляется возможность создания заявки на ремонт и отслеживания ее статуса в режиме реального времени. Замерщики могут просматривать информацию о замере объекта ремонта, а дизайнеры - создавать дизайн-проекты и визуализации. Строителям доступен функционал для предоставления фотоотчета.

Также каждый пользователь приложения имеет доступ к интерфейсу аутентификации, а также клиент имеет еще и регистрацию.

1.3.3. Профили пользователей

Каждый тип пользователя имеет свой профиль в системе. Всего в системе есть 5 типов пользователей. Тип менеджер уже зарегистрирован в системе, менеджер регистрирует в системе новых пользователей таких типов как строитель, дизайнер и замерщик. Клиент регистрируется в системе сам.

Каждый профиль имеет свой набор функциональных возможностей, соответствующих ролям и обязанностям пользователя. Это обеспечивает быстрый и удобный доступ к необходимым функциям и повышает эффективность работы каждого типа пользователей в системе [2].

1.3.4. Ключевые потребности пользователей

Основные потребности всех пользователей являются контроль и учет выполнения заказов клиентов, а также автоматизация процессов.

1.4. Краткий обзор программного продукта

1.4.1. Контекст использования системы

ПП используется в рамках взаимодействия клиентов компании и ее сотрудников. Клиенты могут создавать заявки на ремонт в личном кабинете, а сотрудники компании используют систему для обработки и управления заказами, ввода замеров и данных о состоянии ремонта, общения с клиентами. Программный продукт позволяет значительно ускорить и упростить процесс управления заказами и обеспечить более быстрое и качественное обслуживание клиентов.

1.4.2. Сводка возможностей

Создаваемый ПП позволяет:

- Создание и обработка заявок на ремонт от клиентов
- Назначать ответственных за заказ и контролировать процесс его выполнения
- Вести учет материалов и работ
- Предоставить фотоотчет о состоянии ремонта
- Ввод данных путем измерения и создания дизайн-проектов
- Организовывать эффективное взаимодействие сотрудников компании и клиентов
- Повысить производительность и эффективность компании

ПП позволяет сократить время обработки заявок и управления бизнес-процессами, повысить качество обслуживания клиентов и повысить прибыльность компании.

1.4.3. Предложения и зависимости

Для полноценной работы программного продукта необходимо наличие компьютера с операционной системой Windows 10 или более поздней версии, а также установленным браузером.

Дополнительные предложения, которые могут улучшить работу системы, включают возможность интеграции с платежными системами, учетными системами и системами электронной почты для уведомления клиентов о статусе заказа.

Кроме того, для обеспечения эффективной работы системы и предотвращения сбоев необходимо обеспечить регулярное обновление программно-аппаратных компонентов компьютера.

1.5. Возможности продукта

ПП предоставляет широкий спектр возможностей для автоматизации бизнес-процессов компании по ремонту квартир.

Основные возможности продукта:

- Создание заявок на ремонт квартир в личном кабинете клиента
- Обработка заявок менеджером и назначение ответственных исполнителей
- Внесение данных по замеру объекта ремонта замерщиком
- Создание дизайна и визуализации объекта ремонта дизайнером
- Предоставление фотоотчета по состоянию ремонта строителем

- Управление проектами ремонта и контроль выполнения работ
- Ведение учета затрат и расходов на ремонт квартир

Продукт предоставляет интуитивно понятный и удобный интерфейс для работы с системой. Он основан на принципах модульности и масштабируемости, что позволяет настраивать и расширять функционал продукта в соответствии с потребностями компании.

1.6. Ограничения

В процессе изучения проблемы были выявлены следующие ограничения:

- Ограниченный функционал в первой версии: в первую версию продукта включены только основные функции, необходимые для автоматизации бизнес-процессов компании по ремонту квартир. Некоторые дополнительные функции, которые могут быть полезны для пользователей, могут быть добавлены в будущих версиях продукта.
- Ограниченная поддержка пользователей: в настоящее время поддержка пользователей будет осуществляться только по телефону или электронной почте. В будущем возможно расширение возможностей поддержки, включая чат-поддержку.
- Требования к техническому обеспечению: программный продукт требует наличия определенного технического обеспечения для работы. Подробные требования будут предоставлены во второй главе «Спецификация требований к программному продукту».
- Локализация: на данный момент продукт будет доступен только на русском языке. Локализация на другие языки может быть реализована в будущем, в зависимости от потребностей пользователей.

1.7. Показатели качества

1.7.1. Применимость

Продукт должен быть способен эффективно решать задачи, для которых он был создан, и соответствовать ожиданиям пользователей.

Для обеспечения высокой применимости программного продукта необходимо проводить тщательный анализ требований пользователей и бизнес-процессов компании, на основе которых будет разрабатываться продукт. Также важно проводить тестирование продукта на различных этапах разработки и регулярно выпускать обновления, чтобы улучшить его функциональность и соответствие требованиям пользователей [3].

1.7.2. Надежность

- При сбоях в системе время восстановления работы не должно превышать 1 часа.

- Время, затрачиваемое на обслуживание системы не должно превышать 3% от общего времени работы.

1.8. Старшинство и приоритеты

Первоочередной задачей является разработка функционала, позволяющего эффективно управлять заказами на ремонт квартир, сокращая время на их обработку и увеличивая точность данных.

Далее, важным аспектом является создание удобного пользовательского интерфейса, позволяющего пользователям легко и быстро освоить систему и не тратить много времени на обучение. Также необходимо обеспечить безопасность данных, чтобы избежать утечек конфиденциальной информации.

Приоритетом является обеспечение высокой надежности системы, предотвращение ошибок и сбоев, а также быстрое восстановление работы системы в случае сбоев. Кроме того, важно обеспечить масштабируемость системы и возможность ее расширения в будущем в соответствии с потребностями бизнеса.

Другим важным аспектом является поддержка и обслуживание системы, включая регулярные обновления и исправление ошибок. Все эти аспекты будут учитываться при определении старшинства задач в процессе разработки программного продукта.

1.9. Другие требования к программному продукту

1.9.1. Применяемые стандарты

- ISO 9001:2015 "Системы менеджмента качества - Требования" [4];
- ISO/IEC 12207 "Жизненный цикл программного обеспечения и его процессы" [5];
- ISO/IEC 15504 "Оценка процессов жизненного цикла программного обеспечения (SPICE)" [6];
- ISO/IEC 27001 "Информационная технология. Методы обеспечения информационной безопасности. Системы менеджмента информационной безопасности. Требования" [7];
- OWASP Top 10 "Десять наиболее значимых уязвимостей веб-приложений" [8].

Кроме того, продукт должен соответствовать законодательным требованиям, регулирующим обработку персональных данных и защиту конфиденциальной информации.

1.9.2. Системные требования

- Операционная система: Windows 10 или более поздняя версия, macOS или Linux
- Процессор: Intel Core i3 или выше

- Оперативная память: 4 ГБ или больше
- Свободное место на жестком диске: 250 МБ или больше
- Браузер: Google Chrome, Mozilla Firefox, Microsoft Edge или Safari
- Интернет-соединение: широкополосное соединение с минимальной скоростью 2 Мбит/с

1.9.3. Эксплуатационные требования

- Удобен в использовании для всех категорий пользователей.
- Стабильный и надежный в работе.
- Иметь возможность резервного копирования данных и быстрого восстановления после сбоев.
- Иметь функцию мониторинга и уведомления об ошибках и сбоях в работе.
- Безопасные и защищенные от несанкционированного доступа к данным.
- Обновление и модернизация без остановки работы.
- Иметь документацию, включающую инструкции по установке, настройке и использованию продукта, а также руководство пользователя.

1.10. Требования к документации

1.10.1. Руководство пользователя

Руководство пользователя должно содержать подробную информацию о том, как пользоваться программным продуктом. В нем должны быть описаны все функции и возможности продукта, а также пошаговые инструкции по использованию каждой из них. Документ должен быть написан на простом и понятном языке, без использования технических терминов, которые могут быть непонятны пользователям.

Также в руководстве пользователя должны быть представлены примеры использования продукта и решения возможных проблем, с которыми могут столкнуться пользователи в процессе работы с программой. Документ должен быть доступен в электронном виде и иметь удобную навигацию по разделам.

1.10.2. Интерактивная справка

Интерактивная справка должна быть разработана с целью обеспечить быстрый доступ пользователя к информации о программном продукте, его функциональности и возможностях. Справка должна содержать полное и понятное описание функций, а также инструкции по использованию всех возможностей программного продукта. Кроме того, в справке должны быть приведены сведения о требованиях к системе, ограничениях, решении возможных проблем и т.д.

1.11. Маркировка и пакетирование

1.11.1. Маркировка

Программный продукт должен иметь маркировку, содержащую следующую информацию:

- наименование продукта;
- версию продукта;
- разработчика;
- дату выпуска;
- информацию о лицензировании;
- системные требования.

1.11.2. Пакетирование

ПП должен быть упакован в единый пакет, содержащий все необходимые компоненты и файлы для его корректной установки и работы. Пакет должен содержать следующую информацию:

- руководство пользователя;
- интерактивную справку;
- необходимые библиотеки и компоненты;
- лицензионное соглашение;
- маркировку продукта.

Глава 2. Спецификация требований к программному продукту

2.1. Введение

2.1.1. Назначение

Документ подробно описывает все внешние зависимости и сценарии поведения разрабатываемой автоматизированной информационной системы (АИС). Документ включает перечень нефункциональных требований, проектных ограничений и других аспектов, необходимых для полного и всестороннего описания всех требований участников к проектному решению.

Документ предназначен для лиц, разрабатывающих и тестирующих программную систему.

Данный документ является достаточным для разработки ПП.

2.1.2. Цели создания системы, решаемые задачи и область применения

2.1.2.1. Краткое описание деятельности заказчика и существующей технологии

Список заказов и данные по ним и клиентам хранятся в excel-таблицах. В существующей деятельности системы участвуют 5 актеров:

- Менеджер – сотрудник компании по ремонту квартир, отвечающий за обработку заказа и назначение ответственных за заказ.
- Клиент – заполняет письменную заявку на ремонт
- Замерщик – сотрудник компании, занимающийся замером объекта ремонта.
- Дизайнер - сотрудник компании, занимающийся созданием дизайна, визуализацией объекта.
- Строитель - сотрудник компании, занимающийся ремонтом и предоставлением фотоотчета по состоянию ремонта.

2.1.2.2. Цели создания системы. Эффекты от ее внедрения

Целью создания системы является автоматизация процессов управления заказами и ремонтными работами компании. В результате внедрения системы ожидается увеличение эффективности работы сотрудников компании, повышение качества обслуживания клиентов, сокращение времени на обработку заказов и улучшение управления ресурсами.

Кроме того, внедрение системы позволит снизить вероятность ошибок и пропусков заказов, повысить точность расчета стоимости ремонтных работ, а также улучшить

контроль за сроками выполнения заказов и качеством работ. В результате, компания сможет укрепить свою репутацию на рынке ремонтных услуг, привлечь больше клиентов и увеличить свою прибыльность.

Показателями эффективности от внедрения можно считать:

- Быстродействие
- Надежное хранение данных
- Полный охват функций, подлежащих автоматизации

2.1.2.3. Назначение системы и область применения

Система предназначена для автоматизации процесса обработки заказов и управления информацией о них в компании по ремонту квартир. Наша система позволит менеджерам быстро и эффективно обрабатывать заказы, назначать ответственных за их выполнение, отслеживать статус заказов и получать информацию о выполнении работ. Кроме того, клиенты смогут удобно подавать заявки на ремонт, а замерщики, дизайнеры и строители получать быстрый доступ к информации о заказах и необходимых материалах. Подробный перечень автоматизируемых функций системы, обеспечивающих реализацию ее назначения, приводится в разделе "Функциональность программного продукта".

Типами пользователей нашей системы являются: менеджеры, клиенты, замерщики, дизайнеры и строители. Каждый из них сможет получать доступ только к той информации, которая необходима для выполнения их задач. Преимуществами использования нашей системы являются повышение эффективности работы, ускорение процесса обработки заказов, улучшение контроля над выполнением работ и повышение удовлетворенности клиентов.

2.1.3. Определения, акронимы и сокращения

Определения приведены в документе «Приложение1. Глоссарий проекта».

2.1.4. Ссылки

При создании документа использовались следующие ссылки, вынесенные в раздел «Список используемой литературы»: [9].

2.1.5. Краткое содержание

Данный документ включает в себя три главных раздела: «Введение», «Общее описание» и «Специфические требования».

Общее описание подробно описывает разрабатываемый программный продукт, его функциональность, пользовательские характеристики, ограничения, предположения и зависимости.

Специфические требования описывают внешние требования к интерфейсам программного продукта, спецификации главных вариантов использования, требования к производительности, логические требования к базе данных, ограничения дизайна, ограничения проектирования, соответствие стандартам, атрибуты программного продукта.

Стоит отметить, что Спецификация требований является ключевым документом на этапе разработки программного продукта. Он служит основой для кодирования, тестирования и сопровождения программного продукта. Кроме того, этот документ является основным средством коммуникации между Заказчиком и исполнителем.

2.2.Общее описание

2.2.1. Позиционирование программного продукта

Для функционирования система должна использовать следующие интерфейсы.

Коммуникационные интерфейсы:

- Поточковый сокет TCP/IP.

Пользовательский интерфейс – Web-приложение.

Интерфейс системы должен:

- быть рассчитан на пользователей, не имеющих специальных технических знаний и навыков в области компьютерной техники, и быть легко осваиваем ими.
- обеспечивать интуитивно понятное представление структуры размещенной информации, быстрый и логичный переход к соответствующим разделам системы.
- элементы интерфейса должны быть адаптированы под разрешения различных экранов.

Время, необходимое на освоение системы пользователями не должно превышать 5 минут.

2.2.2. Функциональность продукта

Система должна выполнять следующие функции:

- Авторизация пользователей
- Создание заказов
- Обработка заказов
- Назначение исполнителей для задач

- Отслеживание статуса задач
- Отображение отчетов по выполненным задачам и проделанной работе
- Просмотр статуса заказа в личном кабинете

2.2.3. Характеристики пользователей

Система будет разрабатываться для 5 групп пользователей:

- Менеджер – сотрудник компании по ремонту квартир, отвечающий за обработку заказа и назначение ответственных за заказ;
- Клиент – создает заявку на ремонт в личном кабинете;
- Замерщик – сотрудник компании, занимающийся замером объекта ремонта и внесением данных по замеру;
- Дизайнер - сотрудник компании, занимающийся созданием дизайна, визуализацией объекта;
- Строитель - сотрудник компании, занимающийся предоставлением фотоотчета по состоянию ремонта.

2.2.4. Ограничения

Правовые вопросы: система должна соответствовать действующему законодательству, включая защиту персональных данных пользователей.

Аппаратные ограничения: система должна работать на компьютерах и мобильных устройствах современных версий операционных систем и поддерживать необходимые браузеры.

Интерфейсы с другими приложениями: система может интегрироваться с платежными системами и электронной почтой для отправки уведомлений.

Функции управления: система должна иметь функциональность для управления пользователями, правами доступа и ролями.

Критичность приложения: система должна быть надежной и устойчивой к сбоям, чтобы минимизировать простои и потери данных.

Соображения безопасности и секретности: система должна иметь защиту от несанкционированного доступа, а также должна обеспечивать безопасное хранение и передачу данных.

Также одним из ограничений является доступ к интернету всех пользователей системы.

2.2.5. Предложения и зависимости

Для нормального функционирования системы необходимо:

- Стабильное подключение к сети Интернет.
- Браузер
- Соблюдение всех юридических и правовых требований, касающихся работы с данными клиентов.
- Работа системы должна соответствовать стандартам безопасности данных, установленным для компании.
- Система должна иметь возможность интеграции с другими внутренними системами компании, если это потребуется в будущем.

2.2.6. Распределения требований

Управление заказами: требования, связанные с возможностью принимать, отслеживать и управлять заказами на ремонт квартир. Включает функции создания новых заказов, назначения рабочей бригады, планирования и отслеживания выполнения работ.

Планирование: возможность планирования работ по ремонту квартир на определенные даты.

Последующие версии могут включать следующий функционал:

- Модуль управления заказами: возможность авторизации и регистрации пользователей через социальные сети, публикация отзывов и рекомендаций на страницах социальных сетей.
- Добавление новых типов объектов для ремонта: расширение списка объектов для ремонта, таких как офисные помещения, магазины и т.д.

2.3. Специфические требования

2.3.1. Требования к внешним интерфейсам

2.3.1.1. Пользовательские интерфейсы

- Графический интерфейс пользователя (GUI), представленный в виде веб-приложения, доступен через браузер.
- Форма создания заявки на ремонт, включающая поля для заполнения с информацией о желаемом виде ремонта, адресе объекта и т.д.
- Панель управления для менеджера, позволяющая просматривать все заявки, изменять их статус и присваивать ответственных.

- Интерфейс для замерщика, позволяющий просмотреть детали заявки, внести информацию о замере и просмотреть уже внесенные данные о замерах.
- Интерфейс для дизайнера, позволяющий просмотреть детали заявки, создать дизайн и визуализировать объект ремонта.
- Интерфейс для строителя, позволяющий просмотреть детали заявки, просмотреть фотоотчет по текущему состоянию ремонта.

2.3.1.2. Аппаратные интерфейсы

- Система должна быть доступна через любой современный веб-браузер, такой как Google Chrome, Mozilla Firefox, Safari и др.
- Клиентские компьютеры должны иметь доступ к Интернету.

2.3.1.3. Коммуникационные интерфейсы

- Система должна обеспечивать безопасную передачу данных между сервером и клиентами посредством HTTPS-протокола.
- Система должна поддерживать возможность множественных параллельных подключений клиентов через Интернет.

2.3.2. Функциональные требования

Функциональные требования определяются целями и задачами программного модуля. Исходя из анализа предметной области выявлены актеры, которые будут взаимодействовать с программным продуктом. В таблице 1 дается их краткое описание.

Таблица 1. Актеры системы

Актер	Описание вариантов использования
Менеджер	Следит за выполнением текущего плана и задает регулирующее указание ответственному за объект.
Клиент	Просматривает статусы заявки.
Замерщик	Назначение время замера объекта и прикрепляет документ с замерами.
Дизайнер	После получения документа о замере, проектирует дизайн и визуализацию.
Строитель	Загружает отчет.

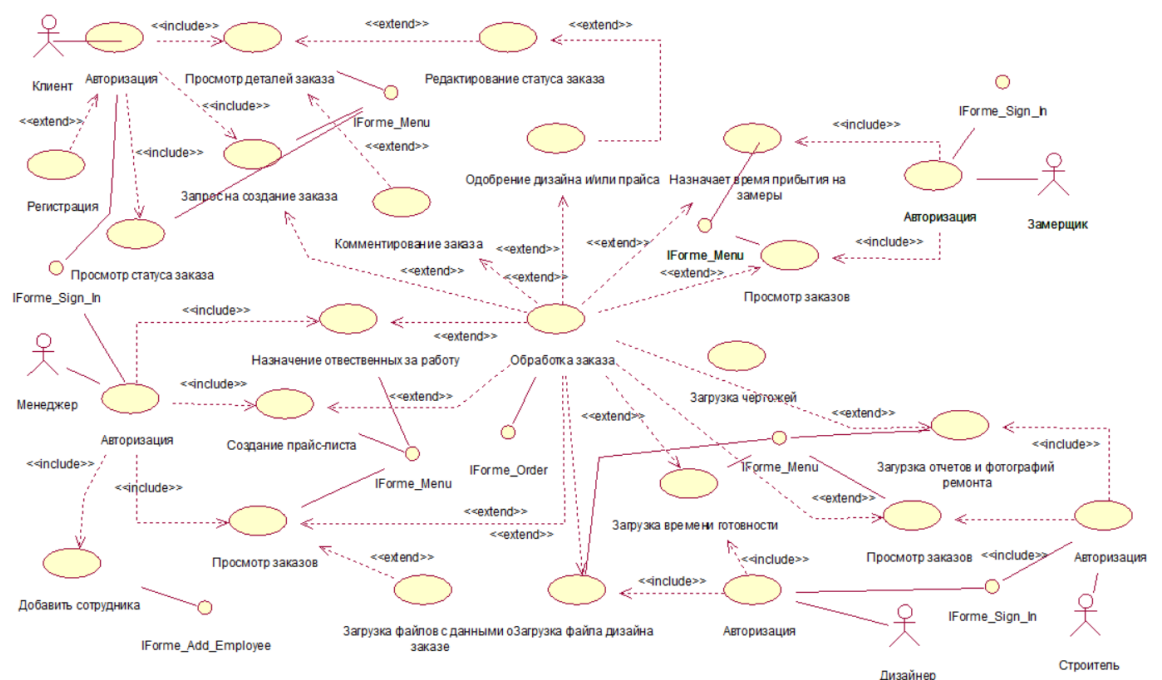


Рисунок 2.1. Диаграмма вариантов использования

Таблица 2. Спецификация вариантов использования.

Основной актер	КОД	Наименование	Формулировка
Клиент	K1	Запрос на создание заказа	Пользователь заполняет форму отправки запроса на заказ
	K1	Просмотр деталей заказа	Пользователь может просматривать такие детали заказа как: информация о заказе, просмотр времени прибытия замерщика, просмотр времени готовности дизайна, просмотр дизайна, просмотр акта, просмотр отчетов о строительстве
	K2	Редактирование статуса заказа	Пользователь может утвердить или отклонить дизайн, акт
	K4	Комментирование заказа	Пользователь может оставить комментарии под актом
	K5	Просмотр статуса заказа	Просмотр статуса заказа в личном кабинете
	K6	Регистрация	Пользователь регистрируется в системе
	K7	Авторизация	Пользователь входит в систему, т.е. вводит логин и пароль
Менеджер	M1	Обработка заказа	Одобрение или отказ в работе
	M2	Назначение ответственных	Назначение ответственных за работу (дизайнер, замерщик, строитель)
	M3	Создание прайс-листа	Создание прайс-листа с ценами за работу и материал
	M4	Просмотр заказа	Пользователь может просматривать детали заказа, его статус и т.д.

	M5	Загрузка файлов с данными о заказе	Пользователь загружает в систему договор, акт о выполненной работе
	M6	Авторизация	Пользователь входит в систему, т.е. вводит логин и пароль
Замерщик	31	Обработка заказа	Пользователь назначает время прибытия на замеры, а также загружает чертежи квартиры
	32	Просмотр заказов	Пользователь просматривает детали заказа
	33	Авторизация	Пользователь входит в систему, т.е. вводит логин и пароль
Дизайнер	Д1	Обработка заказа	Пользователь загружает время готовности работы, а после ссылку на дизайн
	Д2	Просмотр заказов	Пользователь просматривает детали заказа
	Д3	Авторизация	Пользователь входит в систему, т.е. вводит логин и пароль
Строитель	С1	Обработка заказа	Пользователь загружает фотографии хода ремонта, а также отчет о проделанной работе
	С2	Просмотр заказов	Пользователь просматривает детали заказа
	С3	Авторизация	Пользователь входит в систему, т.е. вводит логин и пароль

Анализ сформулированных вариантов использования показал, что с точки зрения потенциальных рисков и архитектурной значимости наиболее существенными являются прецеденты, связанные с работой менеджера и клиента.

Для дальнейшей детализации выбраны три прецедента:

- K1. Запрос на заказ;
- M1. Обработка заказа;
- M2. Назначение ответственных.

Прецедент C1: Запрос на заказ

Запрос на заказ

Краткое описание

Клиент создает запрос на заказ в личном кабинете указывая обязательные пункты. Действующие лица этого прецедента – Клиент.

Поток событий

Прецедент начинается, когда Клиент открывает модуль «Создать заявку» и заполняет форму отправки.

Базовый поток – Запрос на заказ

1. При правильных данных пользователь входит в личный кабинет и переходит в модуль «Создать заявку».
2. Клиент вводит адрес квартиры.
3. Клиент выбирает серию квартиры (SELECT).
4. Клиент вводит количество комнат в квартире.
5. Клиент заполняет дополнительную форму, связанную с подробным описанием комнат, где есть такие поля как название комнаты и ее описание.
6. Клиент прикрепляет картинки квартиры.
7. Клиент нажимает на кнопку «Отправить».
8. Система выводит сообщение о том, что клиент сделал заказ.
9. Система сохраняет все данные.

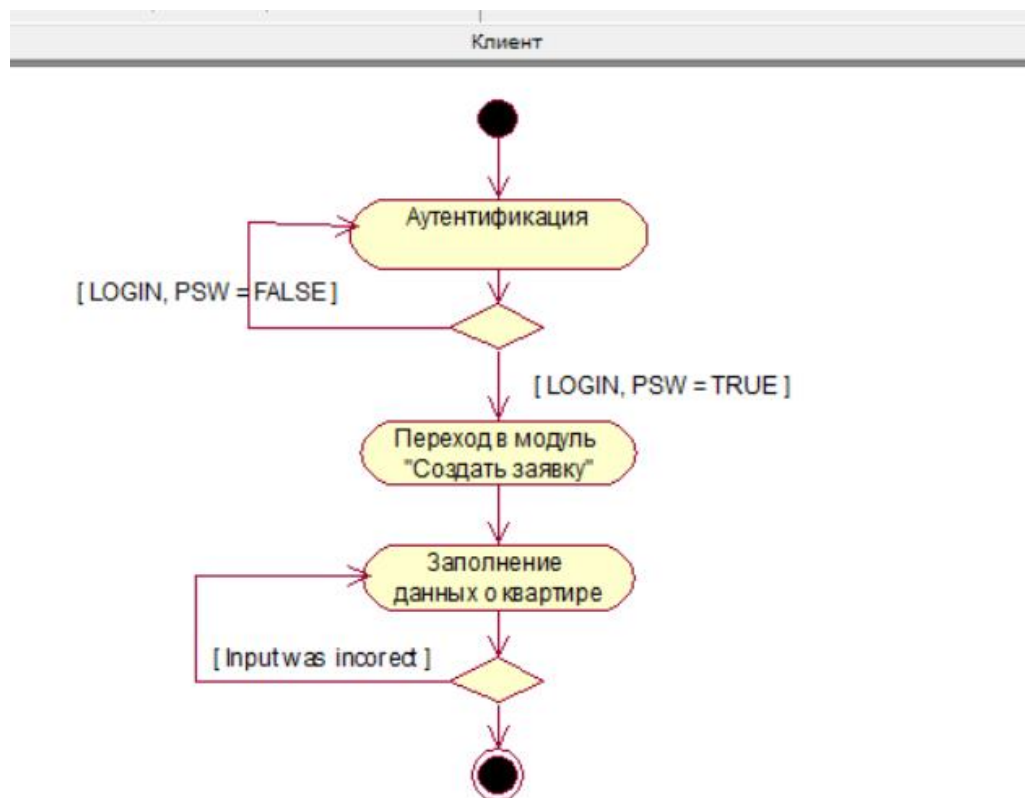


Рисунок 2.2. Диаграмма деятельности прецедента «Создать заказ»

Альтернативные потоки

Если заявка была заполнена некорректна, система просит ввести данные повторно.

Предусловия

Перед тем как начинается этот прецедент, клиент должен быть зарегистрирован в системе.

Постусловия

При успешном окончании прецедента клиент может следить за статусом обработки его заказа в личном кабинете открыв модуль «Все заказы».

Прецедент M1: Обработка заказа

Обработка заказа

Краткое описание

После получения заказа на ремонт Менеджер переходит в модуль «Обработать заказ».

Действующие лица этого прецедента – Менеджер.

Поток событий

Прецедент начинается, когда Менеджер открывает модуль «Обработать заказ» и принимает решение по этому заказу заполняя форму.

Базовый поток – Обработка заказа

1. Менеджер принимает решение по заказу просмотрев данные о квартире которые предоставил клиент
2. Если Менеджер, проанализировав данные решит сделать заказ отказным то вводит причину отказа и нажимает на кнопку «Отказать»
3. Если Менеджер, проанализировав данные решит одобрить заказ то нажимает на кнопку Принять
4. Система сохраняет все данные
5. После решения менеджера клиенту приходит уведомление с результатом обработки

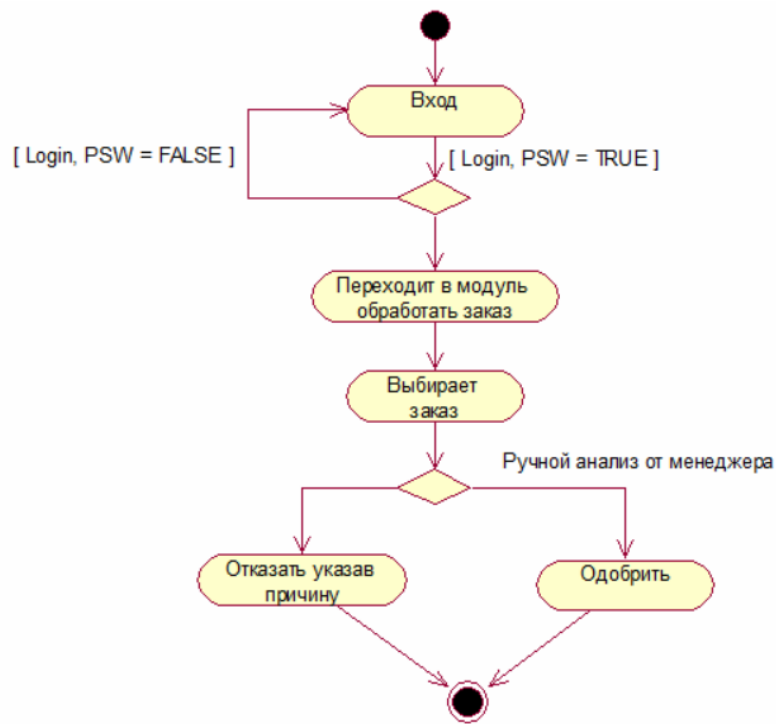


Рисунок 2.3. Диаграмма деятельности прецедента «Обработка заказа» пользователя Менеджер

Альтернативные потоки

Обработка заказа невозможна из-за ошибки сохранения данных о заказе

Если данные по какой-то причине не сохранились, то Менеджер имеет возможность отказать в заказе

Предусловия

Перед тем как начинается этот прецедент, менеджер зарегистрирован в системе.

Постусловия

При успешном окончании прецедента клиент может следить за статусом обработки его заказа в личном кабинете открыв модуль «Все заказы».

Прецедент M2: Назначение ответственных

Назначение ответственных

Краткое описание

После одобрения заказа от менеджера, переходит на страницу о подробной информации о заказе и заполняет форму указывая ответственных за заказ

Действующие лица этого прецедента – Менеджер.

Поток событий

Прецедент начинается, когда Менеджер открывает модуль подробная информация о заказе и назначает ответственных

Базовый поток – Назначение ответственных

1. Переходит в модуль подробная информация о заказе у одобренного заказа
2. Назначает чертежника в форме со списком всех чертежников
3. Назначает Дизайнера в форме со списком всех дизайнеров
4. Назначает строителей в форме со списком всех строителей
5. Нажимает на кнопку Отправить
6. Система сохраняет данные
7. Клиент получает уведомление о назначении ответственных за проект

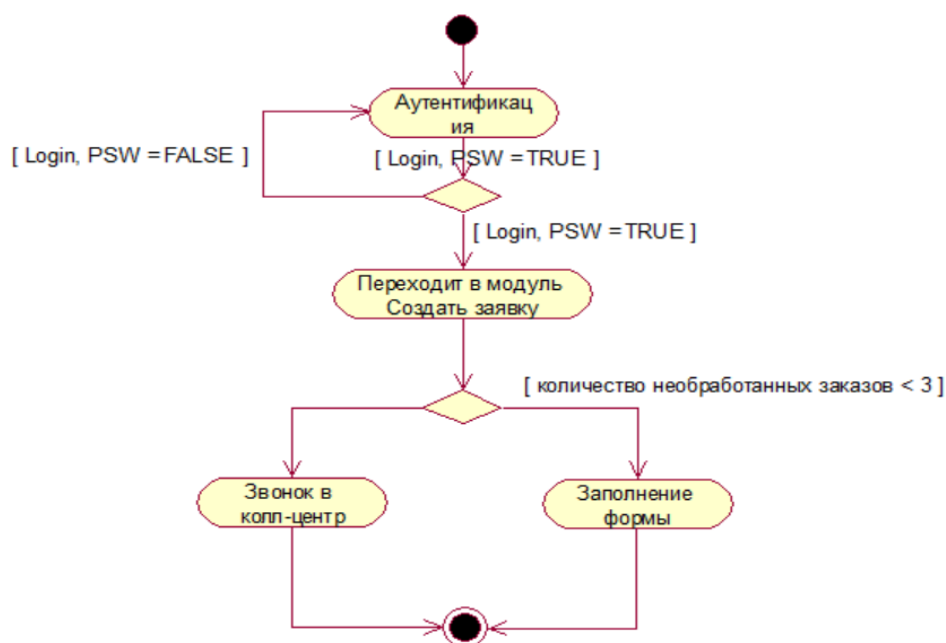


Рисунок 2.4. Диаграмма деятельности прецедента «Назначение ответственных»

Альтернативные потоки

Назначение ответственных невозможно из-за отсутствия списка работников

Если лист со списком работников невозможно получить с базы, то менеджер связывается с колл-центром для оповещения об ошибке

Предусловия

Перед тем как начинается этот прецедент, менеджер зарегистрирован в системе.

Постусловия

При успешном окончании прецедента клиент может следить за статусом обработки его заказа в личном кабинете открыв модуль «Все заказы».

2.3.3. Требования к производительности

Для системы, статические численные требования включают:

- Число поддерживаемых пользователей: система должна поддерживать не менее 100 пользователей одновременно;
- Число поддерживаемых терминалов: система должна поддерживать не менее 50 терминалов одновременно;
- Объем и тип обрабатываемой информации: система должна обрабатывать информацию о заказах, пользователях, исполнителях, работе по заказам, отчетах и другой необходимой информации.

Динамические численные требования включают:

- Число заказов: система должна обрабатывать не менее 50 заказов в день;
- Число одновременных пользователей: система должна поддерживать не менее 10 пользователей, работающих одновременно;
- Время ответа системы: 95% запросов должны обрабатываться менее чем за 2 секунды в нормальных условиях, и менее чем за 5 секунд в пиковые периоды;
- Объем данных: система должна поддерживать обработку не менее 1 Гб данных в день.

2.3.4. Логические требования к базе данных

- База данных должна соответствовать предметной области, каждому объекту предметной области должны соответствовать данные в памяти ЭВМ, а каждому процессу –процедуры обработки данных.
- Данные должны храниться в форме реляционных таблиц.

- Данные должны быть приведены к III нормальной форме.
- Время выполнения транзакции на запрос не должно превышать 15 секунд.
- Время выполнения транзакции на обновление данных не должно превышать 5 секунд.
- СУБД должна выдерживать до 200 транзакций в час.
- СУБД должна обеспечивать безопасность данных, т.е. их целостность и защиту.

2.3.5. Ограничения проектирования

Формулировки условий, модифицирующих требования и интерфейс программы должен быть представлен на русском языке.

2.3.6. Атрибуты программной системы

2.3.6.1. Надежность

- При сбое в системе время восстановления работы не должно превышать 1 часа.
- Время, затрачиваемое на обслуживание системы не должно превышать 3% от общего времени работы.

2.3.6.2. Доступность

- Доступ к функционалу возможен только при доступе к сети интернет.
- Доступ должен предоставляться круглосуточно, 7 дней в неделю.

2.3.6.3. Безопасность

- Система должна хранить логин и историю работы программной системы за последние 7 дней. Протоколироваться должны любые сбои в программной системе и изменения в БД.
- Доступ к функционалу различных групп пользователей разграничивается посредством использования различных модулей в клиентских приложениях каждой из них.
- Связь между клиентскими приложениями происходит только через сервер информационной системы, что позволяет ограничить коммуникации и увеличить их безопасность.
- Система должна быть защищена от SQL-инъекций.

2.3.6.4. Поддерживаемость

- Система должна позволять изменение интерфейса программы в короткие сроки, без изменения функциональной составляющей программы.

2.3.6.5. Переносимость

- Использование стандартизированных протоколов и интерфейсов;
- Использование стандартных форматов данных;
- Использование операционно-независимых библиотек;

- Учет ограничений, связанных с переносимостью, на всех этапах разработки программного обеспечения, включая тестирование и сопровождение;
- Применение методов и инструментов, которые облегчают переносимость программного обеспечения, таких как виртуализация, контейнеризация и т.д.

Глава 3. Проектирование и конструирование ПО

3.1. Введение

Для эффективной работы организаций необходимо решать проблему управления данными. Автоматизация позволяет хранить и структурировать большие объемы данных, что является важной частью функционирования предприятий. В этой связи, важно освоить принципы построения и эффективного применения технологий и программных продуктов, таких как системы управления базами данных, CASE-системы автоматизации проектирования, средства администрирования и другие.

Большинство CASE-средств, основанных на методах структурного или объектно-ориентированного анализа и проектирования, используют спецификации в виде диаграмм или текстов для описания внешних требований, связей между моделями системы, динамики поведения системы и архитектуры программных средств [10].

Для проектирования информационной системы выбрана методология объектно-ориентированного анализа и проектирования на языке UML, что позволяет использовать современные стандарты и подходы в разработке программного обеспечения.

В главе смоделирована диаграмма классов, а также на ее основе диаграмма компонентов и диаграммы последовательности.

3.2. Разработка диаграммы классов

Диаграмма классов (рис. 3.1.) определяет типы классов системы и склонности к общению, которые возникают между ними. На диаграммах классов нанесены также атрибуты классов, операции классов и ограничения, которые устанавливаются на связи между классами. Вид и интерпретация классов существенно зависят от точки зрения (уровня абстракции) [11].

В моделируемой системе определены 7 классов:

- Родительский класс User
- Client
- Manager
- Measurer
- Designer
- Builder
- Order

Также представлены следующие интерфейсы:

- ClientController
- ManagerController
- MeasurerController
- DesignerController
- BuilderController
- OrderController

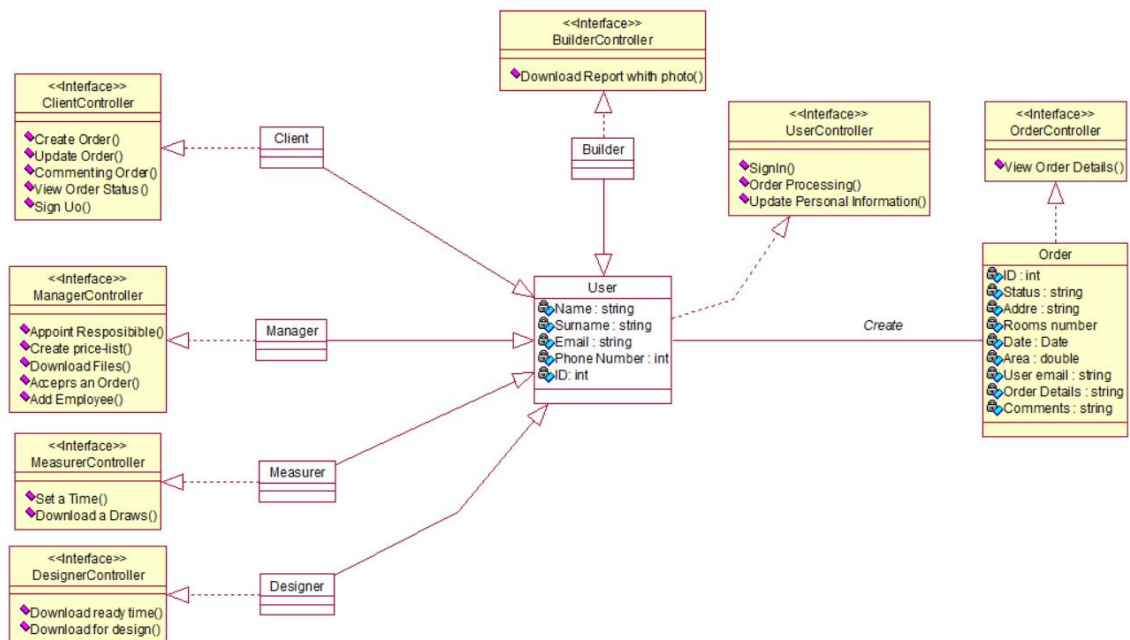


Рисунок 3.1. Диаграмма классов.

3.2.1. Разработка класса User

Client – это родительский класс, от которого наследуются все классы типов пользователей. Класс отвечает за реализацию функционала личной странички пользователя.

Атрибуты класса User:

- ID: int – Уникальный номер в таблице
- Name: string – Имя пользователя
- Surname: string – Фамилия пользователя
- Email: string – Адрес электронной почты
- Phone Number: int – Номер телефона

Методы интерфейса UserController класса User:

- Order Processing() – Обработка заказа
- Sign In() – Авторизация пользователя
- Update Personal Information() – Изменения персональной информации

3.2.2. Разработка класса Order

Order – класс, отвечающий за хранение данных о заказе.

Атрибуты класса Order:

- ID: int – Уникальный номер в таблице
- Status: string – Статус заказа
- Address: string - Адрес
- Rooms: number – Количество комнат
- Date: date - Дата
- Area: double - Площадь
- User email: - Адрес электронный почты пользователя
- Order Details: string – Детали заказа
- Comments: string – Комментарии к заказу

Методы интерфейса OrderController класса Order:

- View Order Details() – Просмотр деталей заказа

3.2.3. Разработка класса Builder

Bulder – класс, отвечающий за обработку заказа пользователя Builder.

Методы интерфейса BuilderController класса Builder:

- Download Report with photos – Загрузка отчетов с фотографиями

3.2.4. Разработка класса Client

Client – класс, отвечающий за обработку заказа пользователя Client.

Методы интерфейса ClientController класса Client:

- Create Order() – Создание заказа
- Update Order() – Обновление данных в заказе
- Commenting Order() – Оставление комментариев к заказу
- View Order Status() – Просмотр статуса заказа
- Sugn Up() - Регистрация

3.2.5. Разработка класса Manager

Manager – класс, отвечающий за обработку заказа пользователя Manager.

Методы интерфейса ManagerController класса Manager:

- AppointResponsible() – Назначение ответственных
- CreatePrice-list() – Создание прайс-листа
- DownloadFiles() – Загрузка файлов
- AcceptsAnOrder() – Принятие заказа
- AddEmployee() – Добавить сотрудника

3.2.6. Разработка класса Measurer

Measurer – класс, отвечающий за обработку заказа пользователя Measurer.

Методы интерфейса MeasurerController класса Measurer:

- SetATime() – Назначение времени замеров
- DownloadADraws() – Загрузка чертежей

3.2.7. Разработка класса Designer

Designer – класс, отвечающий за обработку заказа пользователя designer.

Методы интерфейса DesignerController класса Designer:

- DownloadReadyTime() – Загрузка времени завершения разработки дизайна
- DownloadDesign() – Загрузка дизайна

3.3. Разработка диаграммы компонентов

Диаграммы компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы [11].

При разработке диаграммы компонентов были выявлены следующие программные компоненты:

- Users – пользовательская среда для контактирования с предоставленным системным функционалом.
- WEB-Service – сервер для формирования запросов.
- Server – сервер для обработки запросов
- Remont API – сервер приложения (предоставляющий данные о заказах).
- Django ORM – инструмент для взаимодействия с базой данных

- DataBase – база данных.

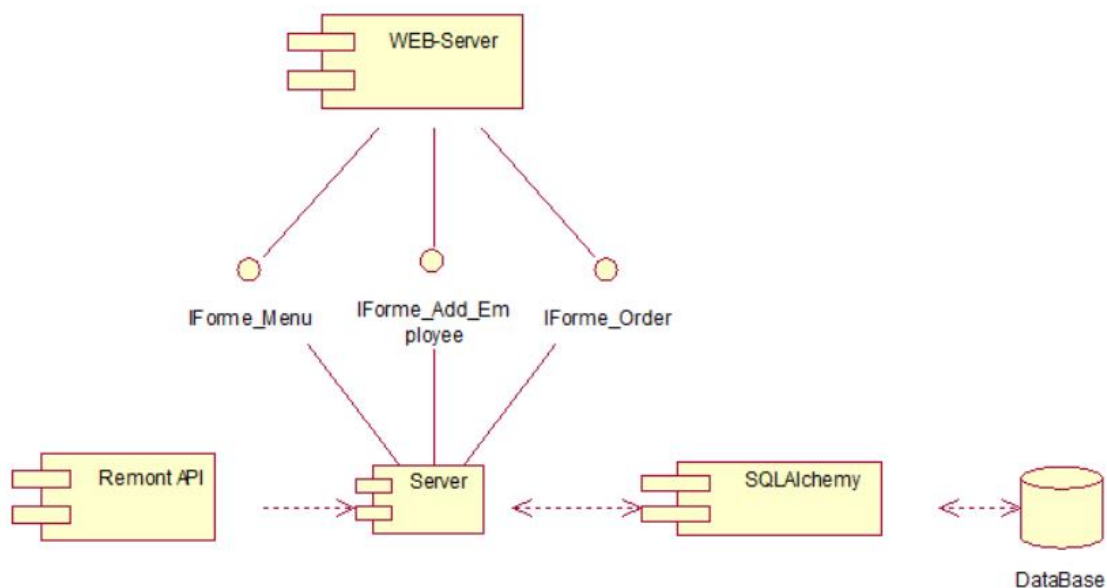


Рисунок 3.2. Диаграмма компонентов.

Исходя из рисунка 3.2. можно сделать вывод, что в ПС будет использоваться трехуровневая архитектура, т.к. на диаграмме четко выражаются три типа компонентов: пользовательский интерфейс, сервер приложений и сервер баз данных.

3.4. Разработка диаграмм последовательностей

Диаграмма последовательности предназначена для моделирования отношений между объектами (ролями, классами, компонентами) Системы в рамках одного прецедента.

Данный вид диаграмм отражает следующие аспекты проектируемой Системы:

- обмен сообщениями между объектами (в том числе в рамках обмена сообщениями со сторонними Системами)
- ограничения, накладываемые на взаимодействие объектов
- события, инициирующие взаимодействия объектов [12].

3.4.1. Разработка диаграммы последовательности сценария

«Авторизация» для всех типов пользователей

При входе в систему пользователь указывает email и пароль. ПС систему отправляет запрос в базу данных на проверку, при успешном вводе, система отображает главную страницу, и пользователь получает доступ к функционалу, который доступ его типу

пользователя. Если же пароль или email были введены неверно, ПС выдает сообщение об ошибке и пользователю предлагают ввести данные повторно.

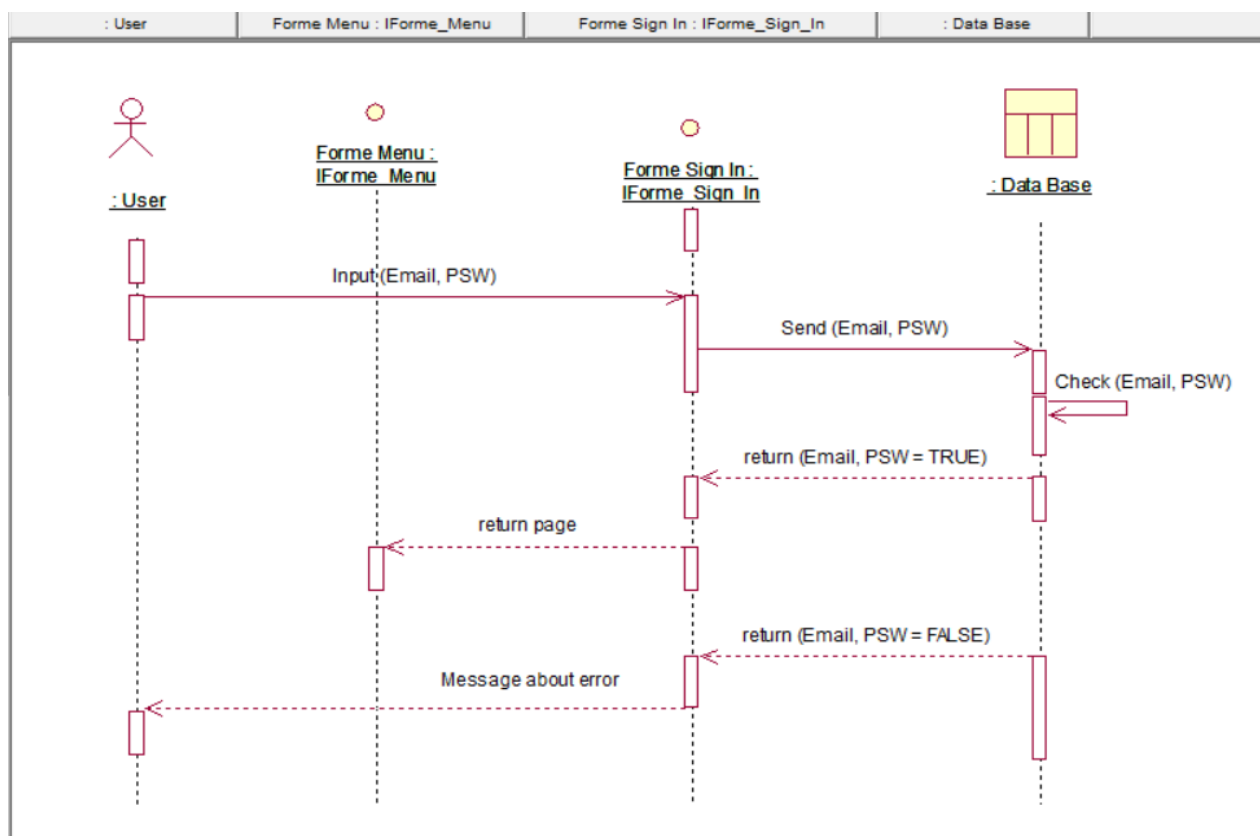


Рисунок 3.4. Диаграмма последовательности сценария «Авторизация».

3.4.2. Разработка диаграммы последовательности сценария «Добавить сотрудника» пользователя менеджер

Пользователь – менеджер вводит данные сотрудника (почту, пароль) и выбирает должность сотрудника из предложенного (строитель, дизайнер, замерщик). Если данные заполнены неверно или пользователь не выбрал должность, ПС выдает сообщение об ошибке. Если же все данные введены верно, ПС выдает сообщение о добавлении сотрудника.

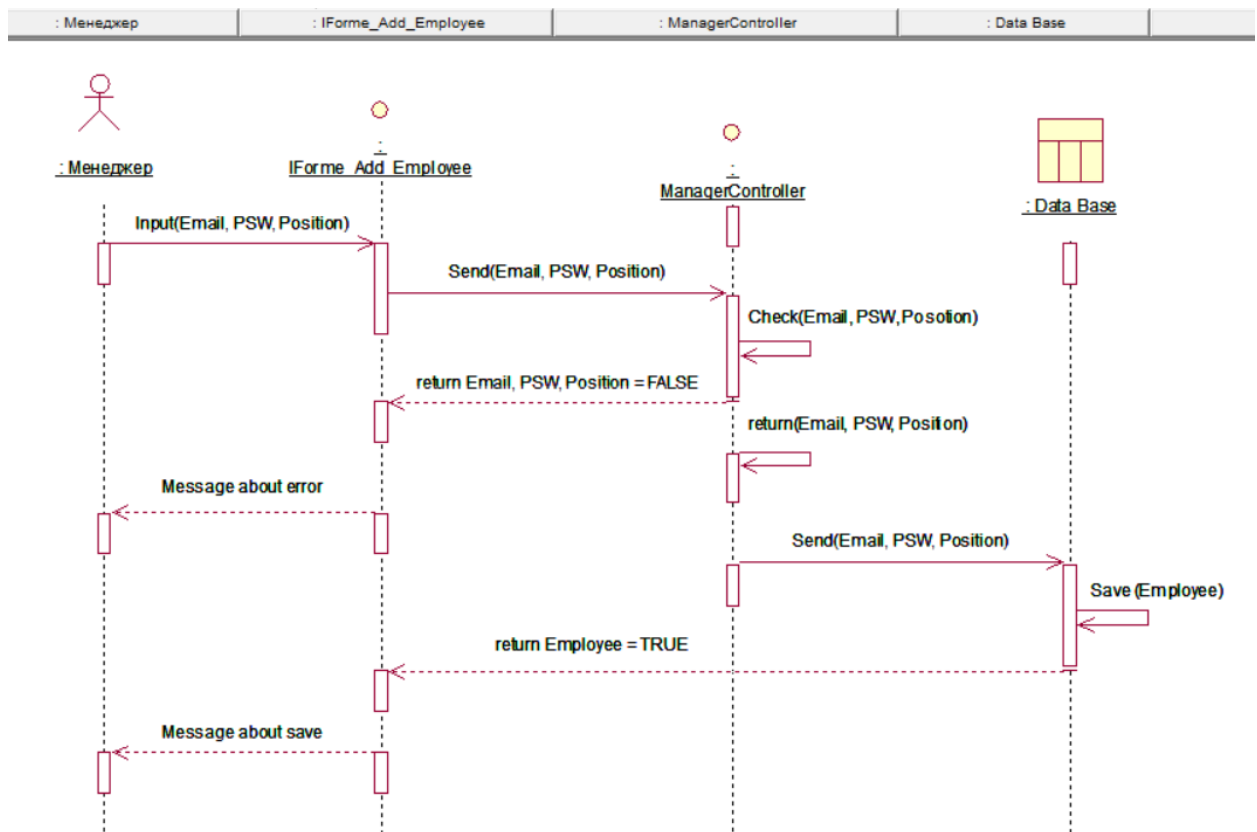


Рисунок 3.5. Диаграмма последовательности сценария «Добавить сотрудника».

3.4.3. Разработка диаграммы последовательности сценария «Запрос на создание заказа» пользователя Клиент

Пользователь – клиент вводит данные заказа. ПС отправляет данные в БД и выдает сообщение о создании запроса на заказ.

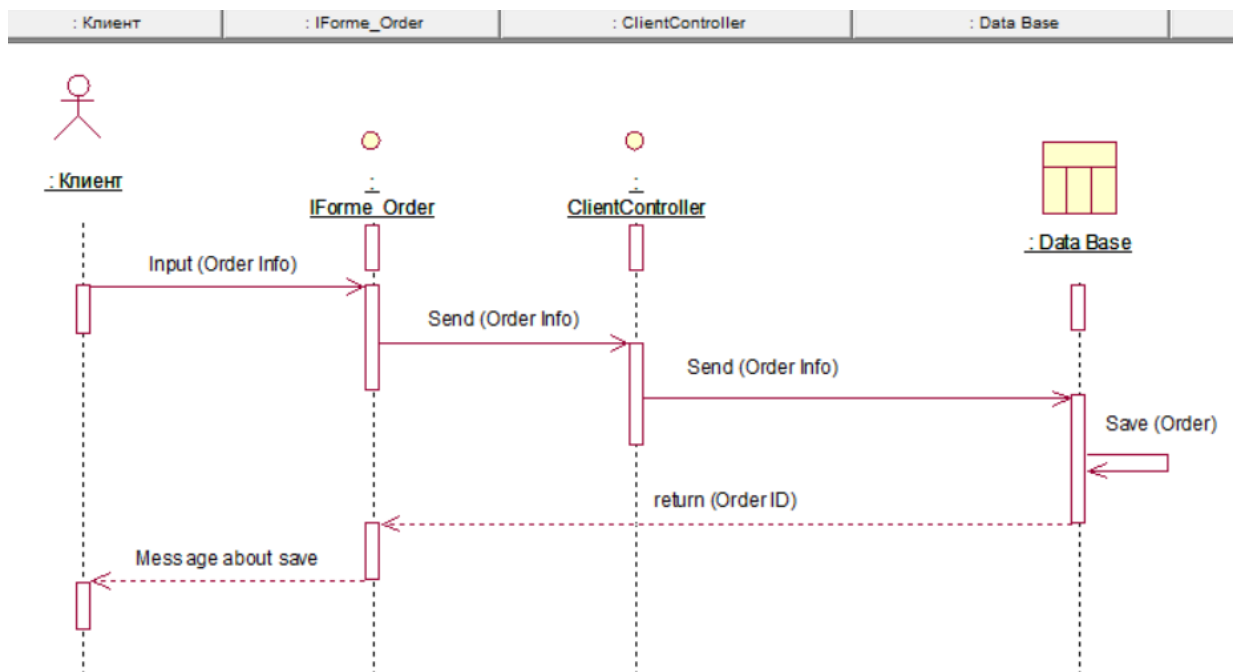


Рисунок 3.6. Диаграмма последовательности сценария «Запрос на создание заказа»
пользователя клиент

3.4.4. Разработка диаграммы последовательности сценариев «Назначение ответственных за работу» и «Создание прайс-листа» пользователя Менеджер

Пользователь - менеджер выбирает ответственных за работу. ПС сохраняет данные в БД.

Менеджер загружает прайс-лист заказа. ПС сохраняет данные в БД.

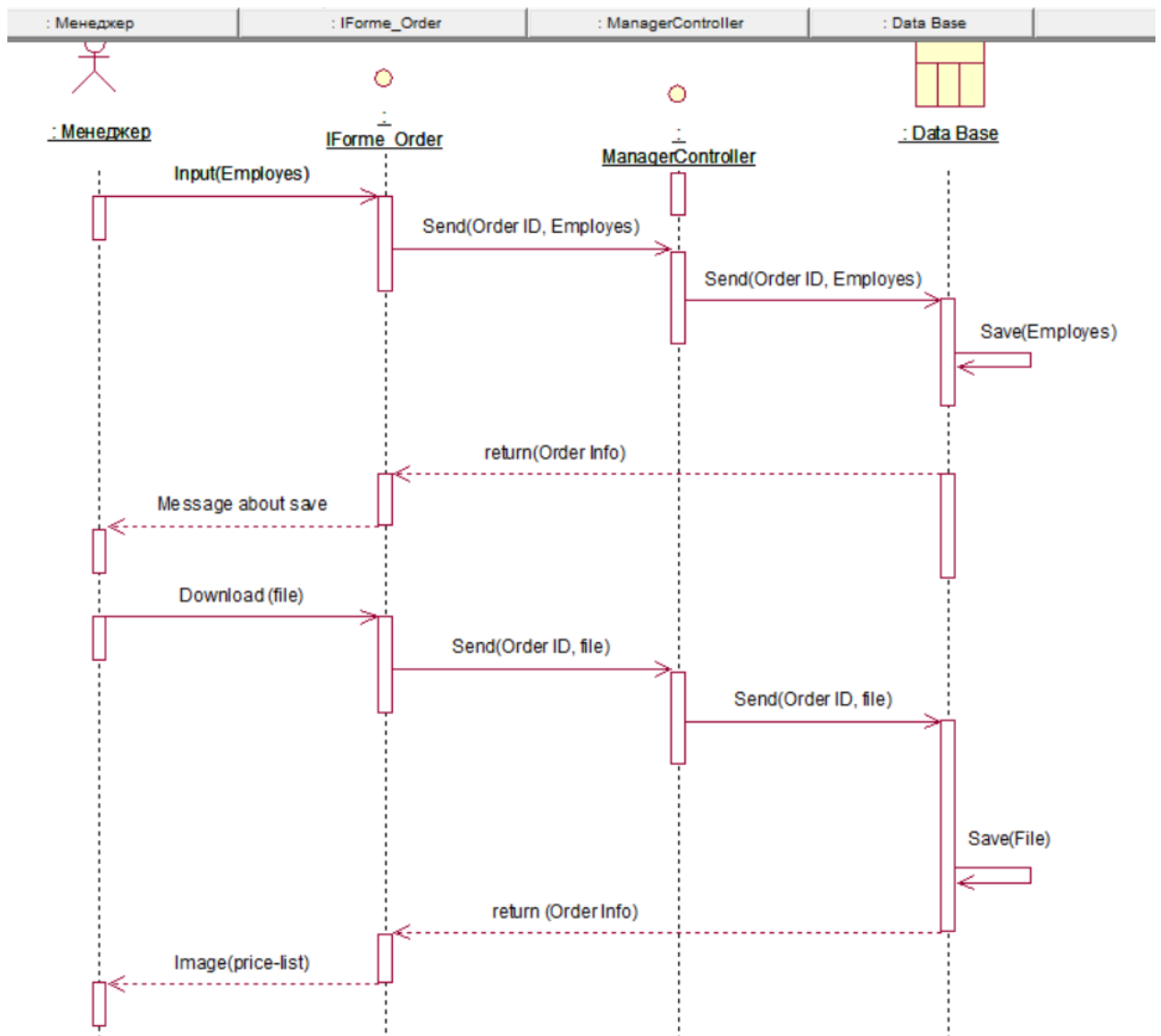


Рисунок 3.7. Диаграмма последовательности сценариев «Назначение ответственных за работу» и «Создание прайс-листа» пользователя Менеджер

3.4.5. Разработка диаграммы последовательности сценариев «Загрузка времени готовности» и «Загрузка файла дизайна»

Пользователь - дизайнер вводит время готовности дизайна. ПС сохраняет данные в БД.

Дизайнер загружает файл дизайна заказа. ПС отправляет данные в БД.

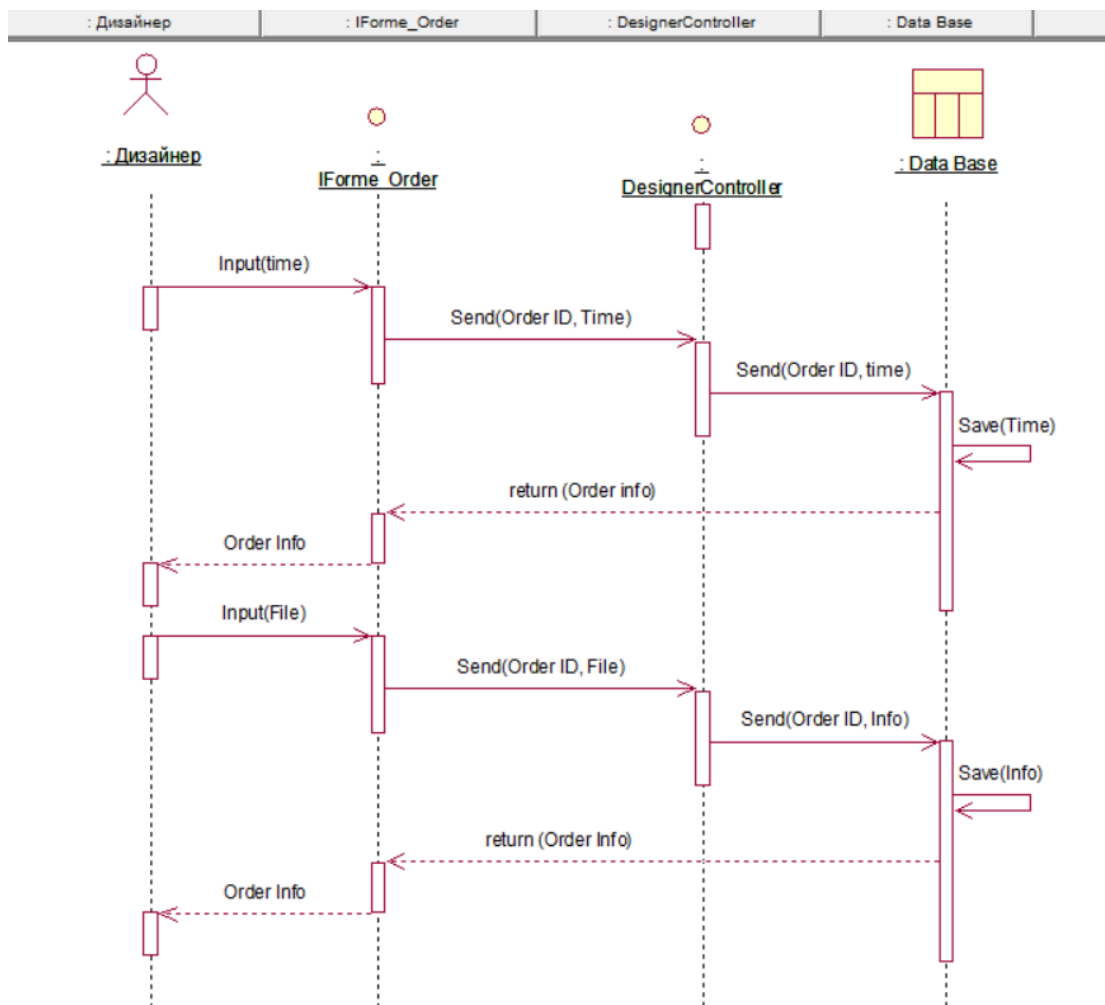


Рисунок 3.8. Диаграмма последовательности сценариев «Загрузка времени заказа» и «Загрузка файла дизайна» пользователя Дизайнер

3.4.6. Разработка диаграммы последовательности сценариев «Назначение времени прибытия на замеры» и «Загрузка замеров» пользователя Замерщик

Пользователь - замерщик вводит время прибытия на замеры. ПС сохраняет данные в БД.

Замерщик загружает файл с чертежами. ПС отправляет данные в БД.

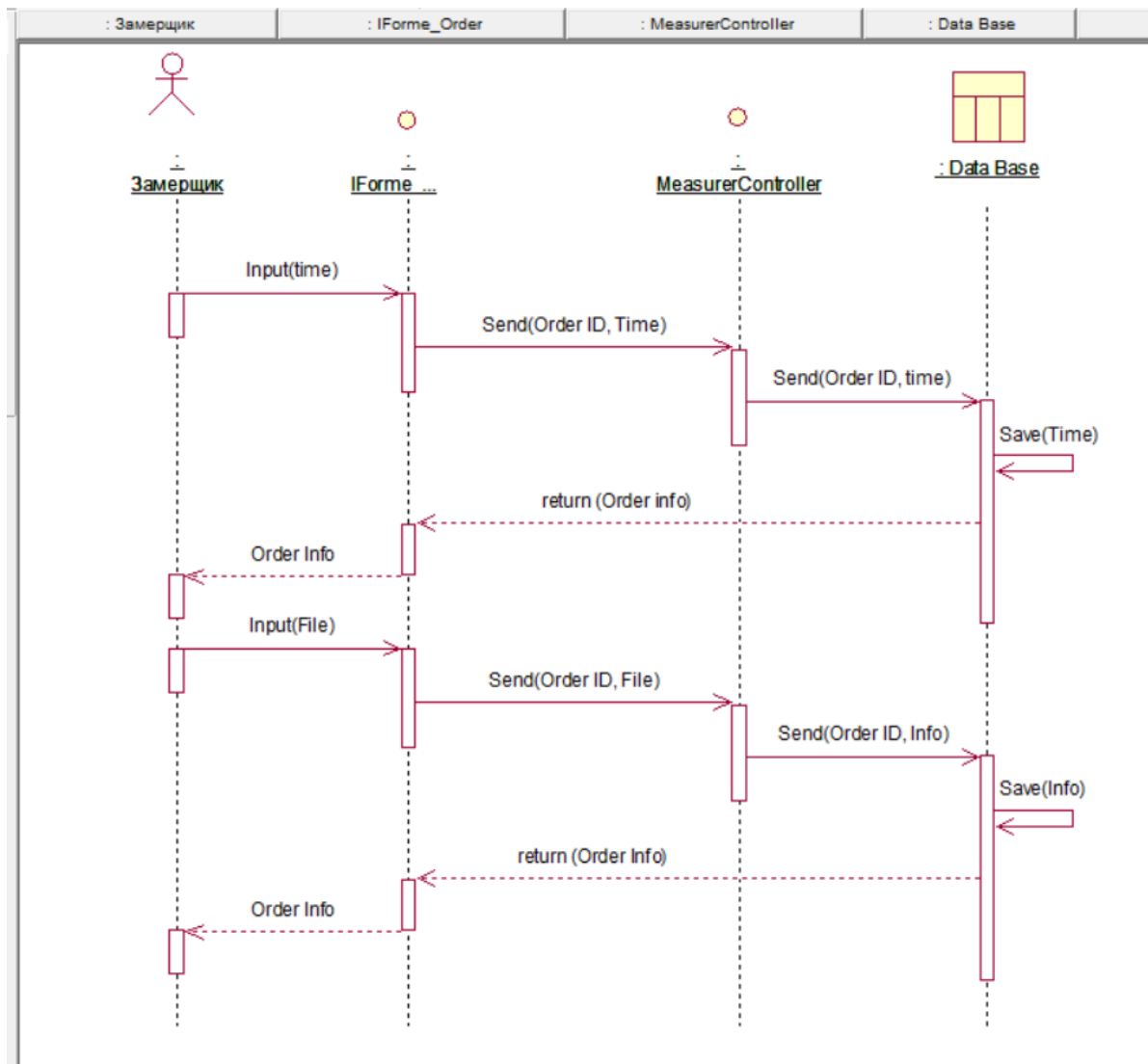


Рисунок 3.9. Диаграмма последовательности сценариев «Назначение времени замеров» и «Загрузка чертежей» пользователя Замерщик

3.4.7. Разработка диаграммы последовательности сценария «Загрузка отчетов и фотографий ремонта» пользователя Строитель

Пользователь – строитель загружает файл с отчетами и фотографиями ремонта. ПС отправляет данные в БД.

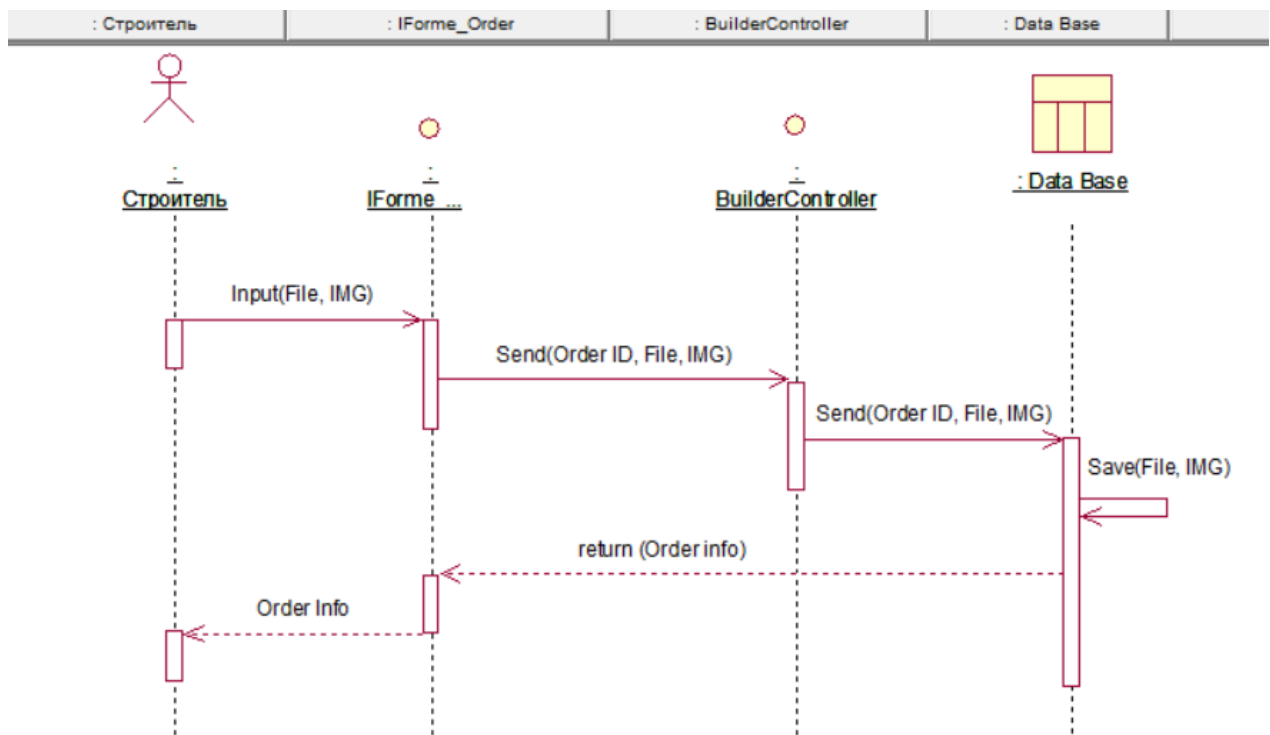


Рисунок 3.10. Диаграмма последовательности сценария «Загрузка отчетов и фотографий ремонта»

Глава 4. Разработка тестов и тестирование ПО

4.1. Введение

4.1.1 Цель

Цель данного плана тестирования - определить деятельности по тестированию и подход к тестированию ПО, которое направлено на автоматизацию бизнес-процессов компании по ремонту квартир.

4.1.2 Область применения

План тестирования охватывает все деятельности по тестированию проекта, включая функциональное тестирование, интеграционное тестирование, производительность и приемочное тестирование пользователем.

4.1.3 Цели

Цели данного плана тестирования:

- Убедиться, что система соответствует заданным функциональным требованиям.
- Оценить производительность системы

4.2. Атрибуты качества

4.2.1 Функциональность

4.2.1.1 Функциональное тестирование

Тест 1: toBase64

Цель: проверить функциональность метода “toBase64”, который преобразует файл в формате указанного расширения в формат base64.

Шаги:

Подготовка тестовых данных:

1. Подготовить тестовый файл в формате, указанном в параметрах метода.
2. Вызвать метод «toBase64» с указанием тестового файла.
3. Проверить, что метод возвращает строку в формате base64.
4. Повторить шаги 1-5 для различных форматов файлов, поддерживаемых методом.

Ожидаемый результат:

- Метод “toBase64” возвращает строку в формате base64, соответствующую содержимому исходного файла.
- Тестовые файлы с различными поддерживаемыми форматами успешно преобразуются в строку base64 без потери данных.

Замечания:

- Для выполнения этого теста потребуются подготовка тестовых файлов разных форматов, например: .txt, .jpg, .png, .pdf, и т.д.

- Проверьте, что метод правильно обрабатывает файлы различных размеров и содержимого.
- Убедитесь, что метод корректно обрабатывает ошибки, например, если указанный файл не существует или невозможно прочитать его содержимое.

Таблица 4.1. Набор тестов

№	File	Что проверяется	Ожидаемый результат	Результат работы программы
1	Png type image	Удачная конвертация	data:image/png;base64,...	data:image/png;base64,...
2	Jpeg type image	Удачная конвертация	data:image/jpeg;base64,...	data:image/ jpeg;base64,...
3	Empty file	Пустой файл	Error message	Error message

Тест 2: compareDates

Цель: проверить функциональность метода “compareDates”, который сравнивает две даты и отдает результат в виде 0,1,2.

Шаги:

1. Подготовка тестовых данных: подготовить две даты на сравнение
2. Вызвать метод «compareDates» с указанием параметров.

Ожидаемый результат:

1. Если date1 больше date2, ожидается возвращение значения 2.
2. Если date1 меньше date2, ожидается возвращение значения 1.
3. Если date1 равна date2, ожидается возвращение значения 0.

Замечания:

В данном шаблоне описан только один функциональный тест. В реальном плане тестирования может-быть несколько тестовых случаев, покрывающих различные сценарии и варианты использования метода compareDates. В описании шагов выполнения и ожидаемого результата можно указать конкретные значения дат для тестирования различных сценариев.

Таблица 4.2. Набор тестов

№	Date 1	Date 2	Что проверяется	Ожидаемый результат	Результат работы программы
1	12.10.23	13.20.21	Date1<Date2	2	2
2	16.05.23	12.05.21	Date1>Date2	1	1
3	15.05.23	15.05.23	Date1=Date2	0	0

Тест 3: getFormattedDate

Цель: Проверить функциональность метода “getFormattedDate”, который сравнивает переводит из формата UTC в обычный формат .

Шаги:

1. Подготовка тестовых данных: подготовить дату для конвертации
2. Вызвать метод «getFormattedDate» с указанием параметра.

Ожидаемый результат:

Ожидается возврат строки, представляющей UTC-дату в dateString.

Замечания:

- В данном шаблоне описан только один функциональный тест. В реальном плане тестирования может-быть несколько тестовых случаев, покрывающих различные варианты формата даты и времени.
- В описании шагов выполнения и ожидаемого результата можно указать конкретные значения UTC-даты и формата для тестирования различных сценариев.

Таблица 4.3. Набор тестов

№	Date	Что проверяется	Ожидаемый результат	Результат работы программы
---	------	-----------------	---------------------	----------------------------

1	2023-04-27T03:13:46.633258	Удачная конвертация	27.04.2023, 09:13:46	27.04.2023, 09:13:46
2	string	Не является датой	Not a date	Not a date

4.2.2 Надежность

4.2.2.1 Тестирование на отказоустойчивость

Ниже приведены примеры тестов на отказоустойчивость для разработки ПО для автоматизации бизнес-процессов компании по ремонту квартир:

Тест на отказ сервера базы данных

Описание

Проверка поведения системы при отказе сервера базы данных.

Шаги выполнения

1. Запустить систему и установить активное подключение к серверу базы данных.
2. Симулировать отказ сервера базы данных путем его выключения или отключения сетевого соединения.
3. Проверить, как система обрабатывает отказ и переключается на резервный сервер базы данных.
4. Убедиться, что система продолжает работать без сбоев и обрабатывает запросы пользователей.
5. Восстановить работу сервера базы данных и проверить, как система возвращается к нормальному режиму работы.

Тест на отказ сервера приложений

Описание

Проверка поведения системы при отказе сервера приложений.

Шаги выполнения

1. Запустить систему и установить активное подключение к серверу приложений.
2. Симулировать отказ сервера приложений путем его выключения или отключения сетевого соединения.
3. Проверить, как система переключается на резервный сервер приложений и продолжает обслуживать запросы пользователей.

4. Убедиться, что функциональность системы не нарушена и пользователи могут успешно выполнять операции.

5. Восстановить работу сервера приложений и проверить, как система возвращается к нормальному режиму работы.

Тест на восстановление после сбоя

Описание

Проверка восстановительных механизмов системы после сбоя.

Шаги выполнения

1. Запустить систему и создать рабочую нагрузку, обрабатывающую запросы пользователей.

2. Симулировать сбой в одном из компонентов системы, например, сервере базы данных или сервере приложений.

3. Проверить, как система обнаруживает сбой и инициирует процесс восстановления.

4. Убедиться, что система успешно восстанавливается и возобновляет обработку запросов.

5. Проверить целостность данных и соответствие ожидаемым результатам операций после восстановления.

Тест на масштабируемость

Описание

Проверка масштабируемости системы при увеличении нагрузки.

Шаги выполнения

1. Запустить систему с базовой нагрузкой и проверить ее производительность.

2. Постепенно увеличивать нагрузку, добавляя дополнительные запросы пользователей или увеличивая объем данных.

3. Мониторить производительность системы и ее способность эффективно обрабатывать увеличенную нагрузку.

4. Убедиться, что система масштабируется горизонтально или вертикально, в зависимости от требований, и продолжает выполнять операции без значительных задержек или ошибок.

5. Проверить, что система возвращается в нормальное состояние после снижения нагрузки.

4.2.3 Тестирование интерфейса

Тест 1: Тестирование модуля входа в систему

Цель: Проверить функциональность модуля входа в систему с помощью полей: почта и пароль.

Шаги:

- Подготовка тестовых данных
- Подготовить валидированные данные для входа
- Вход на определенную страницу URL /auth

Ожидаемый результат:

- Ожидается успешный вход в систему под определенным типом пользователя
- При неправильном вводе ожидается сообщение с ошибкой

Таблица 4.4. Набор тестов

№	Data	Что проверяется	Ожидаемый результат	Результат работы программы
1	Email:pm@example.com Password: admin	Удачный вход как менеджер	Вход в систему	Вход в систему
2	Email:designer@example.com Password: string	Удачный вход как дизайнер	Вход в систему	Вход в систему
3	Email: zamer@example.com Password: string	Удачный вход как замерщик	Вход в систему	Вход в систему
4	Email: user@example.com Password: admin	Удачный вход как клиент	Вход в систему	Вход в систему
5	Email: builder@example.com Password: string	Удачный вход как строитель	Вход в систему	Вход в систему

6	Email: builder@example.com Password: string2	Неверные данные	Сообщение с ошибкой	Сообщение с ошибкой
7	Email: Password:	Пустые данные	Выдаст ошибку об Обязательном поле	Выдаст ошибку об Обязательном поле

Тест 2: Регистрация в систему как клиент

Цель: проверить функциональность модуля регистрация в систему с помощью полей: почта и пароль как клиент

Шаги:

- Подготовка тестовых данных:
- Подготовить данные для входа
- Вход на определенную страницу под URL /auth/registration

Ожидаемый результат:

- Ожидается успешная регистрация в систему под ролью клиент
- При неправильном вводе ожидается сообщение с ошибкой в зависимости от типа ошибки

Таблица 4.5. Набор тестов

№	Data	Что проверяется	Ожидаемый результат	Результат работы программы
1	Email: pm@example.com Password: admin	Регистрация с данными уже существующего аккаунта	Сообщение с ошибкой: Account already exist	Сообщение с ошибкой: Account already exist
2	Email: client56@gmail.com Password: string	Удачный регистрация	Сообщение об удачной регистрации	Сообщение об удачной регистрации

3	Email: Password:	Пустые данные	Выдаст ошибку об обязательном поле	Выдаст ошибку об обязательном поле
---	-------------------------	---------------	--	--

4.3. Подход к тестированию

4.3.1 Стратегия тестирования

Ниже приведены примеры стратегий тестирования для разработки ПО для автоматизации бизнес-процессов компании по ремонту квартир:

4.3.1.1. Стратегия тестирования модулей

Описание

Проверка отдельных модулей системы на корректность их работы.

Цели

- Проверить функциональность каждого модуля системы.
- Выявить дефекты, связанные с отдельными модулями.

Методы

- Тестирование модулей с использованием модульных тестов и инструментов для их автоматизации.
- Тестирование модулей с различными входными данными и проверка выходных результатов.
- Тестирование исключительных ситуаций и обработка ошибок внутри модулей.
- Тестирование взаимодействия между модулями и проверка передачи данных.

Артефакты

- Модульные тесты для каждого модуля системы.
- Отчеты о прохождении модульных тестов и обнаруженных дефектах.

4.3.1.2. Стратегия тестирования интеграции

Описание

Проверка взаимодействия различных компонентов системы и их интеграции.

Цели

- Убедиться, что компоненты системы работают вместе корректно.
- Выявить дефекты, связанные с взаимодействием между компонентами.

Методы

- Тестирование функциональности при интеграции различных компонентов системы.
- Тестирование передачи данных и корректности их обработки между компонентами.

- Тестирование взаимодействия с внешними системами или сервисами.
- Тестирование интерфейсов между компонентами.

Артефакты

- План тестирования интеграции.
- Отчеты о прохождении тестов интеграции и обнаруженных дефектах.

4.3.1.3. Стратегия системного тестирования

Описание

Проверка системы в целом на соответствие требованиям и ожиданиям пользователей.

Цели

- Проверить функциональность системы как единого целого.
- Выявить дефекты, связанные с взаимодействием различных компонентов и функций системы.

Методы

- Тестирование сценариев использования системы, проверка работы ключевых бизнес-процессов.
- Тестирование производительности системы при реалистичных нагрузках.
- Тестирование безопасности и защиты данных.
- Тестирование совместимости с различными операционными системами и браузерами.

Артефакты

- План системного тестирования.
- Отчеты о прохождении системных тестов и обнаруженных дефектах.

4.4.2 Тестовое окружение

4.4.2.1. Frontend тестовое окружение

- Фреймворк разработки: React
- Сборщик: Vite
- Язык программирования: JavaScript/TypeScript
- Тестовый фреймворк: Jest
- Инструмент для UI-тестирования: Cypress
- Дополнительные зависимости и инструменты: Node.js, npm

4.4.2.2. Общие инструменты и зависимости

- Редактор кода: Visual Studio Code
- Командная строка или терминал для выполнения команд тестирования
- Git для контроля версий и управления исходным кодом

- Интеграция с CI/CD Gitlab

4.4.3 Ресурсы

4.4.3.1. Оборудование

- Рабочие станции: компьютеры с достаточными ресурсами для разработки и выполнения тестов.
- Серверы: для развертывания и тестирования серверной части системы, а также базы данных PostgreSQL.
- Сетевое оборудование: для настройки локальных и тестовых сетей, необходимых для тестирования взаимодействия между компонентами системы.

4.4.3.2. Инструменты

- Разработка и тестирование фронтенда: React, Vite, Node.js, Jest, Cypress.
- Разработка и тестирование бэкенда: FastAPI, Python, pytest.
- База данных: PostgreSQL для создания и управления тестовой базой данных.
- Контейнеризация: Docker для создания изолированных тестовых окружений.
- Система контроля версий: Git для управления исходным кодом и совместной работы.
- Интеграционные инструменты: Gitlab CI/CD

4.5. Расписание тестирования

4.5.1 Планирование

4.5.1.1. Планирование и подготовка (Даты: 01.05.2023 - 10.05.2023)

- Определение требований и целей тестирования.
- Составление плана тестирования и его утверждение.
- Подготовка тестового окружения, установка необходимых инструментов и зависимостей.

4.5.1.2. Тестирование модулей (Даты: 11.05.2023 - 20.05.2023)

Проведение модульного тестирования каждого компонента системы.

Выявление и исправление дефектов, связанных с отдельными модулями.

Запуск и автоматизация модульных тестов.

4.5.1.3. Тестирование интеграции (Даты: 21.05.2023 - 31.05.2023)

Проверка взаимодействия между компонентами системы.

Тестирование передачи данных и корректности их обработки.

Проверка взаимодействия с внешними системами или сервисами.

4.5.1.4. Тестирование системы в целом (Даты: 01.06.2023 - 10.06.2023)

Проведение системного тестирования, проверка основных бизнес-процессов.
Тестирование функциональности, производительности и безопасности системы.
Выявление и исправление дефектов, связанных с взаимодействием компонентов.

4.5.2 Управление рисками

4.5.2.1. Риск недостаточного времени для тестирования

План управления риском: предварительно определить необходимые ресурсы и время для тестирования. Составить детальный план тестирования с учетом временных ограничений. Вовремя обнаружить и реагировать на изменения в плане разработки, чтобы убедиться, что достаточно времени выделено для тестирования.

4.5.2.2. Риск недостаточной коммуникации и согласования с командой разработки

План управления риском: установить регулярные коммуникационные каналы с командой разработки. Участвовать в совещаниях, демонстрациях и обсуждениях с командой разработки для понимания изменений, внесенных в систему. Создать механизмы для быстрого обмена информацией и решения обнаруженных проблем с командой разработки.

4.5.2.3. Риск недостаточного покрытия тестами

План управления риском: создать детальный план тестирования, определить критические функциональные области и основные бизнес-процессы, которые требуют тестирования. Убедиться, что все требования и сценарии использования покрываются тестами. Проводить регулярные обзоры тестового покрытия и дополнять тестовую базу при необходимости.

4.5.2.4. Риск недостаточной подготовки тестового окружения

План управления риском: предварительно определить требования к тестовому окружению и его настройку. Создать подробные инструкции по настройке окружения, установке инструментов и зависимостей. Провести проверку и подготовку тестового окружения заранее, чтобы быть уверенным в его готовности к тестированию.

4.5.2.5. Риск несоответствия ожиданиям пользователей

План управления риском: вовлечь бизнес-аналитиков и пользователей в процесс тестирования. Разработать детальные сценарии использования и тестовые случаи, основанные на реальных требованиях

4.6. Критерии завершения

4.6.1 Критерии успешного завершения тестирования

4.6.1.1. Все функциональные требования протестированы

Все функции и возможности системы должны быть протестированы в соответствии с определенными функциональными требованиями. Каждый тестовый случай должен быть выполнен и пройден успешно.

4.6.1.2. Стабильность и отсутствие критических дефектов

Система должна быть стабильной, без критических дефектов, которые могут существенно нарушить работу системы или повлиять на безопасность данных. Все критические дефекты должны быть исправлены или должны быть приняты соответствующие действия для их управления.

4.6.1.3. Уровень покрытия тестами

Достаточное покрытие функциональности системы тестами должно быть достигнуто. Важные бизнес-процессы и ключевые функциональные области должны быть полностью протестированы, и результаты тестирования должны быть задокументированы.

4.6.1.4. Уровень производительности системы

Тестирование производительности должно быть проведено с учетом заявленных требований к производительности системы. Система должна быть способна обрабатывать запросы и выполнять задачи с необходимой скоростью и эффективностью.

4.6.1.5. Соответствие ожиданиям пользователей

Результаты тестирования должны соответствовать ожиданиям пользователей и бизнес-требованиям. Функциональность и возможности системы должны быть протестированы с учетом реальных сценариев использования и требований пользователей.

4.6.1.6. Отчетность и документация

Полная отчетность о проведенном тестировании, включая описание тестовых случаев, результаты тестирования, обнаруженные дефекты и их исправление, должна быть составлена и задокументирована. Все документы и отчеты должны быть готовы к передаче заказчику или заинтересованным сторонам.

4.7. Ограничения и исключения

Ограничения и исключения, которые могут повлиять на проведение тестирования или его результаты в контексте системы автоматизации бизнес-процессов компании по ремонту квартир, включают:

4.7.1. Ограничение доступа к ресурсам

Если тестирование требует доступа к ограниченным ресурсам, таким как база данных с реальными данными клиентов или внешние системы, может потребоваться согласование и настройка прав доступа для тестировщиков. В случае отсутствия такого доступа, некоторые аспекты тестирования могут быть ограничены или требуют замещающих данных или средств.

4.7.2. Ограничения на тестовые данные

Если тестирование требует использования реальных данных или специфических сценариев, могут возникнуть ограничения на доступность таких данных или на их конфиденциальность. В таких случаях может потребоваться создание анонимизированных или фиктивных данных для использования в тестировании.

4.7.3. Исключения функциональности или модулей

В некоторых случаях, определенные функциональные области или модули системы могут быть исключены из тестирования, если они еще находятся в стадии разработки или недоступны для тестирования по другим причинам. В таких случаях, необходимо убедиться, что исключения и их причины документированы и учитываются в общей стратегии тестирования.

4.7.4. Ограничения времени

Ограниченное время может стать препятствием для полного исследования и протестирования всех аспектов системы. В таких ситуациях, требуется определить приоритеты, сконцентрироваться на наиболее критических функциях или сценариях использования, и убедиться, что они тестированы соответствующим образом.

Глава 5. Руководство пользователя

5.1. Назначение системы

Назначение системы для автоматизации бизнес-процессов компании по ремонту квартир заключается в создании централизованной платформы, которая позволяет эффективно управлять всеми аспектами деятельности компании, начиная от управления проектами и заказами, и заканчивая взаимодействием с клиентами и учетом финансовых показателей. Основные цели и назначение такой системы включают:

1. Управление проектами. Система для автоматизации бизнес-процессов позволяет эффективно планировать и контролировать выполнение проектов ремонта квартир. Она предоставляет инструменты для создания заказа, назначения ответственных лиц, отслеживания статуса выполнения заказа, управления ресурсами и контроля сроков.
2. Оптимизация бизнес-процессов: Система помогает оптимизировать бизнес-процессы, связанные с ремонтом квартир, путем автоматизации повторяющихся задач. Это включает прикрепление договоров и актов, отслеживание заказов и их обработка.
3. Управление клиентскими отношениями (CRM): Система обеспечивает управление данными о клиентах, заказах и контактах. Она позволяет хранить и обрабатывать информацию о клиентах, отслеживать историю взаимодействия, управлять запросами и обращениями, а также повышать уровень обслуживания клиентов.
4. Улучшение коммуникации и сотрудничества: Система способствует улучшению коммуникации и сотрудничества внутри компании. Она позволяет сотрудникам взаимодействовать, обмениваться информацией, делиться документацией. В целом, система для автоматизации бизнес-процессов компании по ремонту квартир обеспечивает эффективное управление проектами, повышение качества обслуживания клиентов, оптимизацию бизнес-процессов и улучшение общей производительности компании. Она помогает сократить временные и ресурсные затраты, повысить прозрачность и контроль, а также создает основу для дальнейшего развития и роста компании.

5.2. Условия применения системы

Клиентские устройства: Пользователи будут использовать свои компьютеры, ноутбуки, планшеты или смартфоны для доступа к системе через веб-браузеры. Важно убедиться, что их устройства соответствуют минимальным требованиям для работы с веб-приложениями, такими как поддержка современных веб-браузеров, достаточная

оперативная память и процессорная мощность для выполнения операций на клиентской стороне.

Сетевое оборудование: Для обеспечения стабильного и безопасного соединения между клиентскими устройствами и облачным сервером требуется надежное сетевое оборудование. Это может включать маршрутизаторы, коммутаторы, брандмауэры и прочие сетевые устройства, которые обеспечат надежную передачу данных. Интернет-соединение: Важным компонентом будет качество и стабильность интернет-соединения. Высокоскоростное и стабильное соединение поможет обеспечить быстрый доступ к системе и плавное взаимодействие с ней.

Квалификация пользователя в отношении базовых навыков работы с компьютером включает следующие аспекты:

1. Операционные системы: Пользователь должен быть знаком с одной или несколькими операционными системами, такими как Windows, macOS или Linux. Он должен уметь выполнять основные операции, такие как включение и выключение компьютера, управление файлами и папками, установка и удаление программ.

2. Рабочий стол и интерфейс: Пользователь должен быть знаком с элементами рабочего стола и интерфейса операционной системы. Это включает понимание панели задач, меню Пуск и возможности переключения между приложениями.

3. Веб-браузеры: Пользователь должен знать, как использовать веб-браузеры, такие как Google Chrome, Mozilla Firefox или Microsoft Edge. Он должен уметь открывать новые вкладки, вводить адреса веб-сайтов, осуществлять поиск в Интернете и сохранять закладки.

4. Работа с файлами и папками: Пользователь должен быть знаком с основами работы с файлами и папками. Это включает создание новых папок, перемещение и копирование файлов, переименование и удаление файлов и папок.

5. Офисные приложения: Пользователь должен знать, как использовать базовые функции офисных приложений, таких как Microsoft Word, Excel или Google Документы, Google Таблицы. Он должен уметь создавать и редактировать документы, таблицы, презентации, а также сохранять их в нужном формате.

6. Электронная почта и коммуникация: Пользователь должен знать, как использовать электронную почту и осуществлять коммуникацию через интернет. Это включает создание и отправку электронных писем, прикрепление файлов к письмам и чтение полученных сообщений.

7. Безопасность и защита: Пользователь должен иметь базовое понимание о мерах безопасности в компьютерной среде. Это включает использование паролей, обновление программ и операционной системы, а также осведомленность о потенциальных угрозах, таких как вредоносные программы и фишинг.

5.3. Подготовка системы к работе

Для подготовки системы к работе и запуска веб-приложения необходимо выполнить следующие инструкции:

1. Установите необходимые компоненты:

- Убедитесь, что на компьютере установлена подходящая операционная система (например, Windows, macOS или Linux),
- Убедитесь, что на компьютере установлен веб-браузер (например, Google Chrome, Mozilla Firefox, Microsoft Edge) последней версии.

2. Запустите веб-браузер:

Откройте веб-браузер на вашем компьютере.

3. Введите URL-адрес приложения:

В адресной строке веб-браузера введите URL-адрес приложения “http://localhost:3000”

4. Введите учетные данные:

На странице авторизации введите предоставленные вам учетные данные (логин и пароль) или зарегистрируйтесь.

5. Нажмите на кнопку "Отправить":

После ввода учетных данных нажмите на кнопку " Отправить", чтобы выполнить процесс аутентификации.

6. Подождите загрузки приложения:

Подождите, пока веб-приложение загрузится. Это может занять некоторое время в зависимости от производительности системы и стабильности соединения.

7. Приложение готово к использованию:

После успешной загрузки вы будете перенаправлены на главную страницу. Теперь вы можете начать использовать функциональность приложения.

5.4. Описание операций

5.4.1. Операция «Вход в систему»

1) Условия при соблюдении которых возможно выполнение:

- Учетная запись пользователя должна быть зарегистрирована в системе.
- Предоставленные учетные данные (логин и пароль) должны быть правильными и соответствовать зарегистрированной учетной записи.

2) Подготовительное действие:

- Открытие веб-браузера и переход на страницу авторизации приложения по URL: <http://localhost:3000/auth>.
- Получение учетных данных (логин и пароль) от пользователя.

3) Основные действия:

- Ввести логин и пароль в соответствующие поля на странице авторизации.
- Нажать на кнопку "Отправить", чтобы отправить данные на сервер для проверки.

4) Заключительные действия:

- Ожидание ответа от сервера о результате аутентификации.
- В случае успешного входа в систему, перенаправление на главную страницу и отображение данных о профиле.
- В случае ошибочного ввода данных или неправильных учетных данных, отображение сообщения об ошибке.

5) Ресурсы, расходуемые на операцию:

- Компьютер или мобильное устройство пользователя.
- Веб-браузер для взаимодействия с интерфейсом приложения.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные (логин и пароль), предоставленные пользователем для аутентификации.
- Ресурсы сервера, включая процессорное время и память, для обработки запроса на аутентификацию и проверку данных.

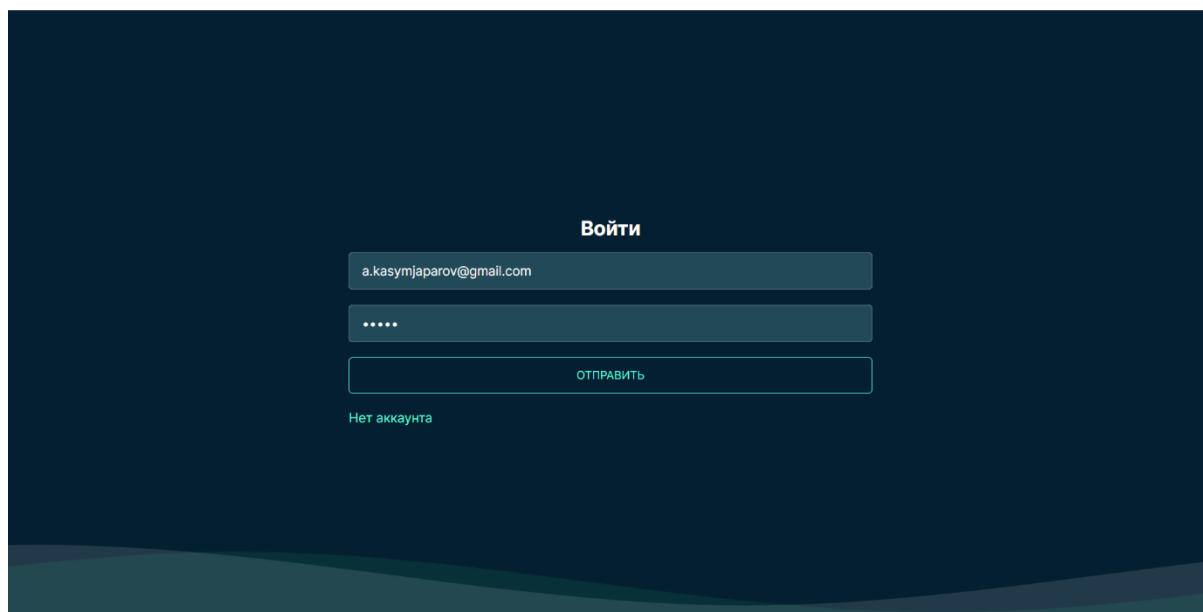


Рисунок 5.1. Операция «Войти в систему»

5.4.2. Операция «Регистрация пользователя»

1) Условия при соблюдении которых возможно выполнение:

- Пользователь должен иметь доступ к интернету.
- Уникальный идентификатор пользователя должен быть доступным и не зарегистрированным в системе.

2) Подготовительное действие:

- Открытие веб-браузера и переход на страницу регистрации приложения по URL: <http://localhost:3000/auth/registration>.
- Подготовка необходимой информации, такой как логин и пароль.

3) Основные действия:

- Ввести запрашиваемую информацию, такую как адрес электронной почты и пароль, в соответствующие поля на странице регистрации.
- Нажать на кнопку "Отправить", чтобы отправить данные на сервер для создания новой учетной записи пользователя.

4) Заключительные действия:

- Ожидание ответа от сервера о результате регистрации.
- В случае успешной регистрации, перенаправление на страницу входа в систему.
- В случае ошибок или неправильно введенных данных, отображение сообщений об ошибках.

5) Ресурсы, расходуемые на операцию:

- Компьютер или мобильное устройство пользователя.
- Веб-браузер для взаимодействия с интерфейсом приложения.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Уникальные идентификаторы пользователя (например, электронная почта) для проверки доступности и регистрации новой учетной записи.
- Ресурсы сервера, включая процессорное время и память, для обработки запроса на регистрацию и создания новой учетной записи.

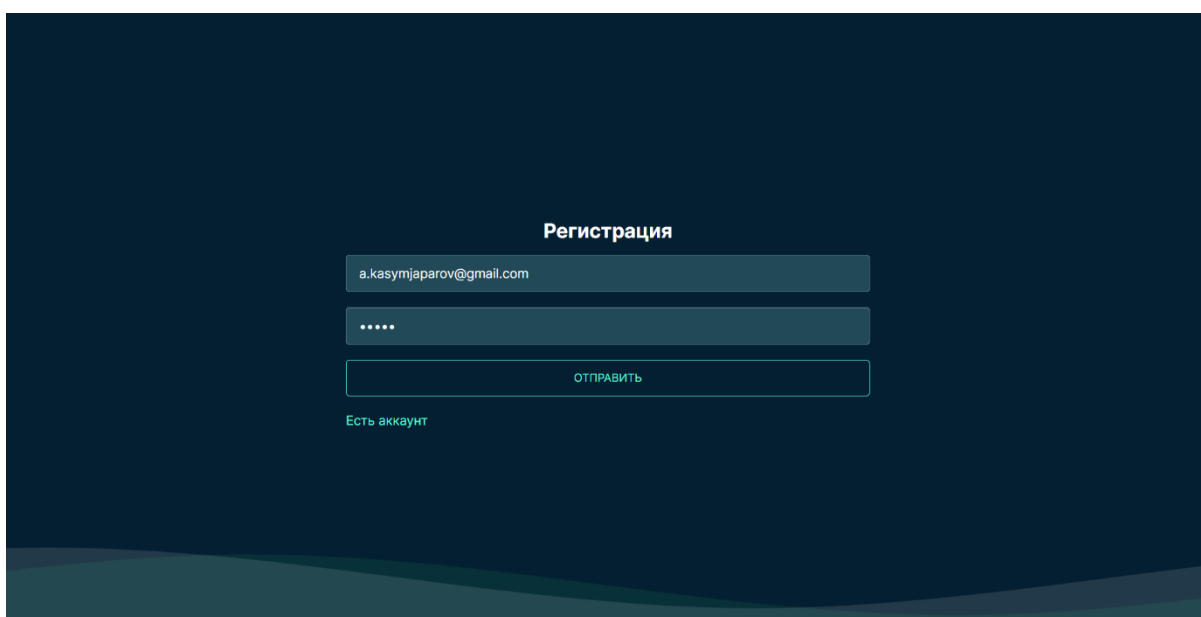


Рисунок 5.2. Операция «Регистрация пользователя»

5.4.3. Операция «Редактирование данных профиля»

1) Условия при соблюдении которых возможно выполнение:

Клиент должен быть зарегистрирован в системе и иметь активную учетную запись.

2) Подготовительное действие:

- Открытие веб-браузера и переход на страницу регистрации приложения по URL: http://localhost:3000/edit_account.
- Подготовка необходимой информации, такой как имя, фамилия и номер телефона.

3) Основные действия:

- Ввести запрашиваемую информацию, такую как имя, фамилия и номер телефона, в соответствующие поля на странице редактировании данных.

- Нажать на кнопку "Отправить", чтобы отправить данные на сервер для редактирования своих данных.

4) Заключительные действия:

- Ожидание ответа от сервера о результате редактирования.
- В случае успешного редактирования, можно увидеть успешное сообщение.
- В случае ошибок или неправильно введенных данных, отображение сообщений об ошибках.

5) Ресурсы, расходуемые на операцию:

- Компьютер или мобильное устройство пользователя.
- Веб-браузер для взаимодействия с интерфейсом приложения.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Ресурсы сервера, включая процессорное время и память, для обработки запроса на редактирования учетной записи.

Рисунок 5.3. Операция «Редактирование данных профиля»

5.4.4. Операция «Создание заказа клиентом»

1) Условия при соблюдении которых возможно выполнение:

- Клиент должен быть зарегистрирован в системе и иметь активную учетную запись.
- Клиент должен редактировать данные своего профиля.

2) Подготовительное действие:

- Вход в систему клиента с использованием учетных данных.
- Подготовка необходимой информации, такой как адрес, количество комнат, серия квартиры, площадь всей квартиры, список комнат с их названиями, проблемами, площадью и фотографиями.

3) Основные действия:

- Переход на соответствующую страницу, где предоставляется возможность создания заказа.
- Ввод информации о заказе, такой как адрес, количество комнат, серия квартиры, площадь всей квартиры, список комнат с их названиями, проблемами, площадью и фотографиями.
- Проверка введенных данных на корректность и полноту.
- Нажатие на кнопку "Отправить" для отправки данных на сервер.

4) Заключительные действия:

- Ожидание ответа от сервера о результате создания заказа.
- В случае успешного создания заказа, отображение подтверждающего сообщения и переход на страницу заказов пользователя.
- В случае ошибок или неправильных данных, отображение сообщений об ошибках и возможность повторного ввода или корректировки данных.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство клиента.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные, предоставленные клиентом для авторизации.
- Ресурсы сервера, включая процессорное время и память, для обработки запроса на создание заказа и сохранения данных заказа в базе данных.

The image shows a web application interface for VKR. On the left is a dark sidebar with navigation links: 'Главная' (Home), 'Сделать заказ' (Make order), and 'Мои заказы' (My orders). The main content area has a title 'Подать заявку' (Submit application) and a subtitle 'Укажите адрес вашей квартиры, количество комнат, серию квартиры, опишите кратко проблемы в ваших комнатах.' (Specify the address of your apartment, the number of rooms, the apartment series, and briefly describe the problems in your rooms). The form contains several input fields: 'Адрес' (Address), 'Количество комнат' (Number of rooms), 'Серия' (Series), 'Площадь всей квартиры' (Total apartment area), 'Комнаты и их проблемы' (Rooms and their problems) with a sub-field 'Название комнаты' (Room name), 'Опишите проблему комнаты' (Describe the room problem), 'Площадь комнаты' (Room area), and 'Выберите фотографии комнаты' (Select room photos) with a sub-field 'ВЫБЕРИТЕ ФОТОГРАФИИ КОМНАТЫ'. At the bottom left is a 'ДОБАВИТЬ КОМНАТУ' button, and at the bottom right is an 'ОТПРАВИТЬ' button.

Рисунок 5.4. Операция «Создание заказа клиентом»

5.4.5. Операция «Просмотр списка своих заказов»

1) Условия при соблюдении которых возможно выполнение:

- Пользователь должен быть авторизован в системе и иметь активную учетную запись.
- У пользователя должны быть соответствующие права доступа для просмотра списка заказов.

2) Подготовительное действие:

Вход в систему пользователя с использованием учетных данных (логин и пароль).

3) Основные действия:

- Переход на страницу списка заказов пользователя по URL: <http://localhost:3000/orders>.
- Есть выбор фильтра статуса заказа, представленного в виде элемента выбора (select) с различными вариантами статусов заказа (например, "В обработке", "Отказано", "Замер закреплен").
- Возможность получить список с пагинацией
- Отправка запроса на сервер для получения списка заказов, относящихся к данному пользователю и соответствующих выбранному фильтру статуса.

4) Заключительные действия:

- Отображение списка заказов, относящихся к пользователю и соответствующих выбранному фильтру статуса.

- Просмотр информации о каждом заказе, такой как Адрес, дата создания, тип ремонта, текущий статус и другие детали заказа.
- Возможность просмотра подробной информации о заказе.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство пользователя.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные (логин и пароль), предоставленные пользователем для авторизации.
- Ресурсы сервера, включая процессорное время и память, для обработки запроса на получение списка заказов и фильтрацию по статусу.

АДРЕС	КОЛИЧЕСТВО КОМНАТ	СТАТУС	ДАТА	ПЛОЩАДЬ	ПОЛЬЗОВАТЕЛЬ
вышкавы	1	В обработке	07.06.2023, 13:45:43	232м²	a.kasymjanov@gmail.com
fydfdf	5	Договор прикреплен	27.04.2023, 09:13:46	120м²	kasymjanov@gmail.com
ddfdfd	6	В обработке	17.04.2023, 11:14:53	1111м²	fff@fdf.ru
Лыжная В	2	В обработке	12.04.2023, 14:00:56	12341234м²	user@example.com
Heheh	4	В обработке	04.04.2023, 23:05:15	7272м²	user@example.com
Жилой жолц	6	Статус изменен заказчиком	04.04.2023, 23:02:55	45м²	user@example.com
dasd	2	В обработке	04.04.2023, 10:30:04	23м²	user2@example.com
68 ул. Калыка Акиева, Башкиев 720001	3	Дата готовности дизайна установлена	03.04.2023, 18:21:30	60м²	talknigov95@gmail.com
lkjhghghghk	2	Отказано	29.03.2023, 19:42:32	123м²	teurem.cs@gmail.com
12 мкрн, 45, 6	3	Статус изменен заказчиком	29.03.2023, 10:05:10	85м²	pochta4@gmail.com

Рисунок 5.5. Операция «Просмотр списка своих заказов»

5.4.6. Операция «Обработка заказа менеджером»

1) Условия при соблюдении которых возможно выполнение:

- Менеджер должен быть авторизован в системе и иметь соответствующие права доступа для обработки заказов.
- Заказ должен находиться в статусе: «В обработке».

2) Подготовительное действие:

- Вход в систему менеджера с использованием учетных данных (логин и пароль).
- Переход на страницу списка заказов, требующих обработки.

3) Основные действия:

- Просмотр списка заказов, требующих обработки.
- Выбор нужного заказа для обработки.
- Ознакомление с деталями заказа, такими как адрес, площадь, количество комнат, серия квартиры, дата создания, статус и прочие сведения.
- Определение решения по заказу: отказ от заказа или его принятие.
- В случае отказа от заказа, указание причины отказа (необходимо заполнить соответствующее поле).
- В случае принятия заказа, выбор ответственных работников: дизайнер, замерщик, строители, которые будут заниматься выполнением проекта и выбор типа ремонта: капитальный и косметический.

4) Заключительные действия:

- Сохранение изменений, сделанных менеджером, связанных с обработкой заказа.
- Обновление статуса заказа в соответствии с принятым решением.
- Отправка уведомлений о принятии или отказе от заказа клиенту.
- Обновление списка заказов, отражающего изменения в статусе и ответственных лицах заказов.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство менеджера.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные (логин и пароль), предоставленные менеджером для авторизации.
- Ресурсы сервера, включая процессорное время и память, для обработки запросов на обновление статуса заказа и отправку уведомлений.

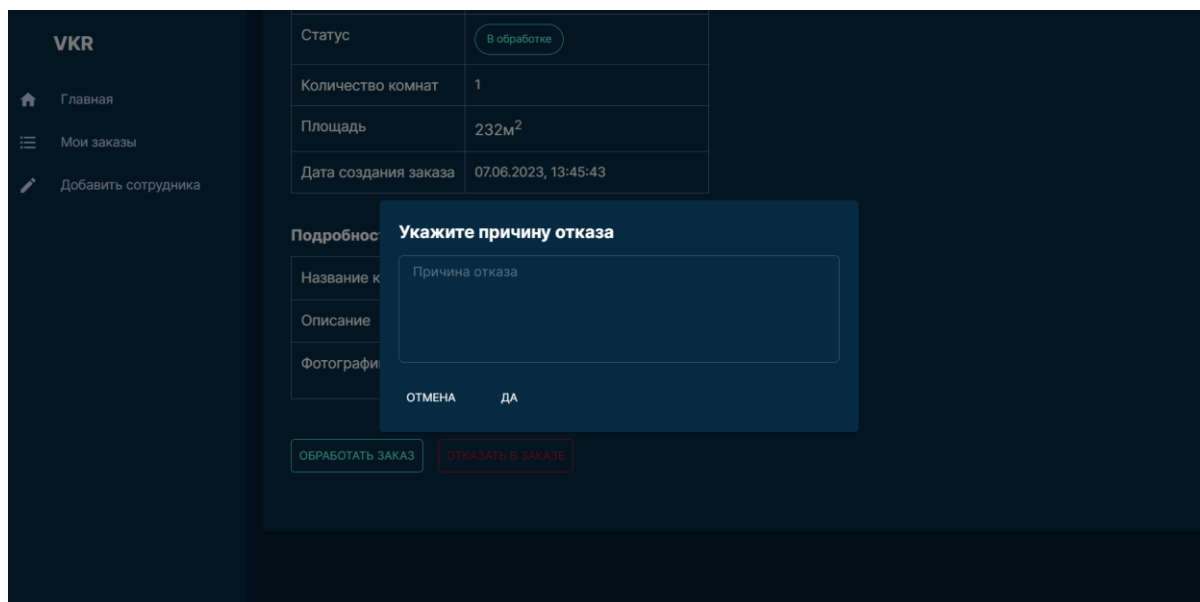


Рисунок 5.6. Операция «Обработка заказа менеджером при отказе»

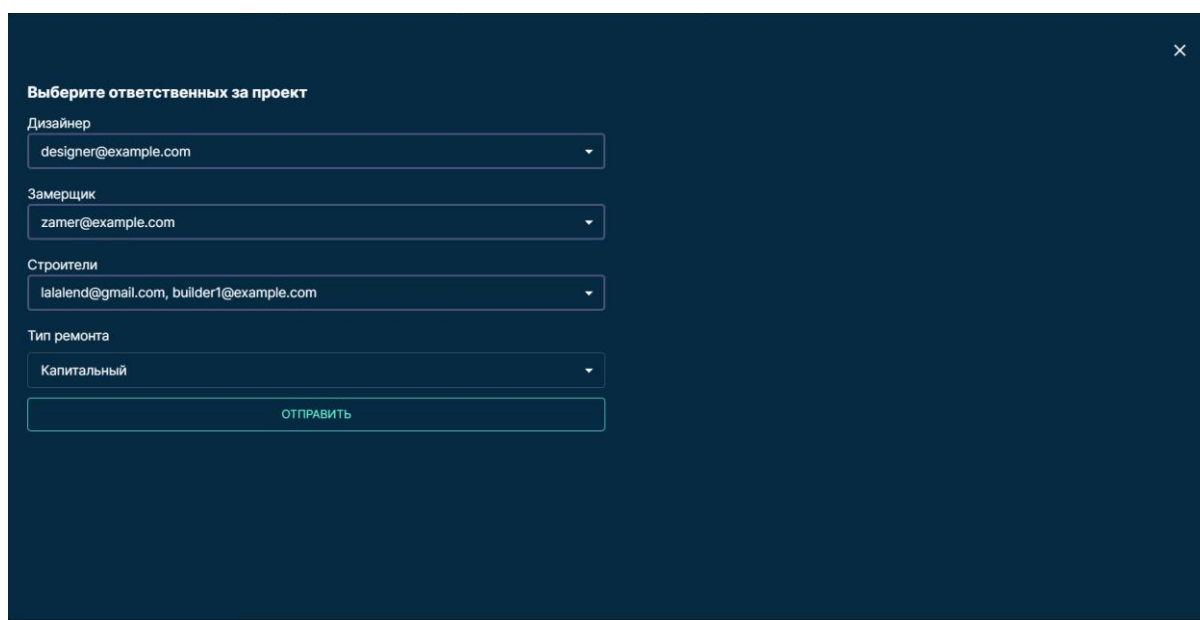


Рисунок 5.7. Операция «Обработка заказа менеджером при принятии заказа»

5.4.7. Операция «Установка времени прихода на объект замерщиком»

1) Условия при соблюдении которых возможно выполнение:

- Замерщик должен быть авторизован в системе и иметь соответствующие права доступа для установки времени прихода.
- Заказ должен быть назначен на замерщика и иметь статус: Ответственные назначены

2) Подготовительное действие:

- Вход в систему замерщика с использованием учетных данных.

- Переход на страницу списка заказов, назначенных на замерщика по URL: <http://localhost:3000/orders>.

3) Основные действия:

- Просмотр списка заказов, назначенных на замерщика.
- Выбор нужного заказа для установки времени прихода.
- Ознакомление с деталями заказа.
- Установка времени прихода на объект, указывая точное время, когда замерщик будет находиться на месте работы.

4) Заключительные действия:

- Сохранение изменений, сделанных замерщиком, связанных с установкой времени прихода на объект.
- Обновление статуса заказа: Время замера установлено
- Уведомление клиенту о назначенном времени прихода замерщика.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство замерщика.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные (логин и пароль), предоставленные замерщиком для авторизации.
- Ресурсы сервера, включая процессорное время и память, для обработки запросов на обновление времени прихода и отправку уведомлений.

Площадь	45м²	Строитель	Почта: builder1@example.com Номер телефона: +996444444444 ФИО: вфывфв фвыфвф
Дата создания заказа	04.04.2023, 23:02:55		

Подробнее по комнатам	
Название комнаты	Так себе квартира на самом деле
Описание	Очень маленькая
Фотографии комнаты	

Выберите дату и время замера

Рисунок 5.8. Операция «Установка времени прихода на объект замерщиком»

5.4.8. Операция «Прикрепление замера»

1) Условия при соблюдении которых возможно выполнение:

- Пользователь под ролью замерщик должен быть авторизован в системе и иметь соответствующие права доступа для прикрепления замера.
- Заказ должен быть создан и находиться в статусе: Время замера установлено

2) Подготовительное действие:

- Вход в систему пользователя с использованием учетных данных.
- Переход на страницу списка заказов или детальной страницы заказа, на которую нужно прикрепить замер.

3) Основные действия:

- Просмотр списка заказов или выбор нужного заказа для прикрепления замера.
- Подготовка необходимой документации и информации, связанной с замером
- Загрузка документа и комментария на страницу, предназначенную для замеров.
- Подтверждение прикрепления замера и сохранение введенной информации.

4) Заключительные действия:

- Обновление статуса заказа, отражающего прикрепление замера.
- Уведомление всех заинтересованных сторон (клиента, исполнителей) о прикрепленном замере.
- Доступ к прикрепленным замерам для дальнейшей работы и планирования ремонта.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство пользователя.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные, предоставленные пользователем для авторизации.
- Ресурсы сервера, включая процессорное время и память, для обработки запросов на прикрепление замера и сохранение информации.

VKR

Главная
Мои заказы

Подробности по комнатам

Название комнаты	Так себе квартира на самом деле
Описание	Очень маленькая
Фотографии комнаты	

Выберите файл замера

ВЫБЕРИТЕ ФАЙЛ ЗАМЕРА

Комментарий

ОТПРАВИТЬ

Данные по замеру

Дата и время приезда на объект	23.06.2023, 14:25:00
Комментарий	
Дата создания замера	07.06.2023, 14:25:33

Рисунок 5.9. Операция «Прикрепление замера»

5.4.9. Операция «Добавление сотрудника»

1) Условия при соблюдении которых возможно выполнение:

- Пользователь должен быть авторизован в системе как менеджер
- Указанная почта сотрудника должна быть уникальной и не привязана к другому аккаунту в системе.

2) Подготовительное действие:

- Вход в систему пользователя с использованием учетных данных.
- Переход на страницу добавления нового сотрудника по URL: http://localhost:3000/add_employees.

3) Основные действия:

- Ввод почты сотрудника в соответствующее поле.
- Ввод пароля для аккаунта сотрудника.
- Выбор указание должности сотрудника.
- Подтверждение добавления сотрудника и сохранение введенных данных.

4) Заключительные действия:

Появление в базе профиля с введенными данными.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство пользователя.

- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.

The screenshot shows a web application interface for VKR. On the left is a dark blue sidebar with a menu containing 'Главная' (Home), 'Мои заказы' (My orders), and 'Добавить сотрудника' (Add employee) with a green plus icon. The main area has a dark blue header with the VKR logo and a hamburger menu icon. Below the header, the title 'Добавить сотрудника' (Add employee) is displayed in white, followed by the instruction 'Укажите почту, пароль и должность сотрудника' (Specify the employee's email, password, and position). The form itself is a light blue box with three input fields: 'Почта' (Email), 'Пароль' (Password), and 'Должность' (Position) which is a dropdown menu. At the bottom of the form is a green button labeled 'ОТПРАВИТЬ' (SEND).

Рисунок 5.10. Операция «Добавление сотрудника»

5.4.10. Операция «Установка времени готовности дизайна»

1) Условия при соблюдении которых возможно выполнение:

- Дизайнер должен быть авторизован в системе и иметь соответствующие права доступа для установки времени прихода.
- Заказ должен быть назначен на дизайнера и иметь статус: Замер закреплен

2) Подготовительное действие:

- Вход в систему дизайнера с использованием учетных данных.
- Переход на страницу списка заказов, назначенных на замерщика по URL: <http://localhost:3000/orders>.

3) Основные действия:

- Просмотр списка заказов, назначенных на замерщика.
- Выбор нужного заказа для установки времени готовности дизайна.
- Ознакомление с деталями заказа.
- Установка времени готовности дизайна, указывая точное время, когда дизайн квартиры будет готов.

4) Заключительные действия:

- Обновление статуса заказа: Дата готовности дизайна установлена
- Уведомление клиенту и менеджеру о назначенном времени готовности дизайна.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство замерщика.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.

5.4.11. Операция «Закрепление дизайна»

1) Условия при соблюдении которых возможно выполнение:

- Пользователь должен быть авторизован в системе как дизайнер.
- Дизайн должен быть разработан и предоставлен дизайнером для закрепления.

2) Подготовительное действие:

- Вход в систему пользователя с использованием учетных данных.
- Получение дизайна от дизайнера в соответствующем формате (например, файлы PDF или MP4).

3) Основные действия:

- Переход на страницу заказа, для которого необходимо закрепить дизайн.
- Загрузка файла визуализации (например, PDF или MP4) на страницу заказа.
- Ввод описания к визуализации, поясняющего особенности и характеристики представленного дизайна.
- Прикрепление списка фотографий дизайна, если таковые имеются, путем загрузки файлов и ввода соответствующего описания к каждому файлу.

4) Заключительные действия:

- Уведомление клиента о закрепленном дизайне и отправка уведомления с соответствующей информацией.
- Ожидание подтверждения клиента по поводу закрепленного дизайна. Если клиент подтверждает дизайн, обновление статуса заказа, отражающего подтвержденный дизайн.
- Если клиент не подтверждает дизайн, возможность повторного закрепления нового дизайна путем повторения операции.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство пользователя.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные (логин и пароль), предоставленные пользователем для авторизации.
- Файлы визуализации дизайна, в соответствующем формате (например, PDF или MP4).
- Описание к визуализации, поясняющее особенности и характеристики представленного дизайна.
- Список фотографий дизайна, включающий файлы и соответствующие описания к каждому файлу.

Прикрепить дизайн

Прикрепите визуализацию, скрины с дизайном и описанием к ним.

Прикрепите визуализацию

ПРИКРЕПИТЕ ВИЗУАЛИЗАЦИЮ

Описание к визуализации

Описание

Фотография № 1

Выберите фотографии дизайна

ВЫБЕРИТЕ ФОТОГРАФИИ ДИЗАЙНА

Описание к фотографии

Описание

ДОБАВИТЬ

ОТПРАВИТЬ

Рисунок 5.11. Операция «Закрепление дизайна»

5.4.12. Операция «Утвердить или отказать дизайн»

1) Условия при соблюдении которых возможно выполнение:

- Клиент должен быть авторизован в системе и иметь соответствующие права доступа для утверждения или отказа дизайна.
- Дизайн должен быть предоставлен клиенту для рассмотрения и принятия решения.

2) Подготовительное действие:

- Вход в систему клиента с использованием учетных данных.

- Получение представленного дизайна.

3) Основные действия:

- Переход на страницу заказа.
- Ознакомление с представленными визуализациями, описанием и фотографиями.
- Принятие решения по дизайну:
- Если дизайн устраивает клиента, выбор опции "Утвердить".
- Если дизайн не устраивает клиента, выбор опции "Отказать" и указание причины отказа (например, несоответствие требованиям, стилистические предпочтения и т. д.).

4) Заключительные действия:

- Отправка решения (утверждение или отказ) и указание причины в систему.
- Обновление статуса заказа в соответствии с принятым решением клиента.
- Уведомление о принятом решении и указанной причине.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство клиента.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные (логин и пароль), предоставленные клиентом для авторизации.
- Представленный дизайн, включая визуализации, описание и фотографии.
- Поле для указания причины отказа, если таковая имеется.

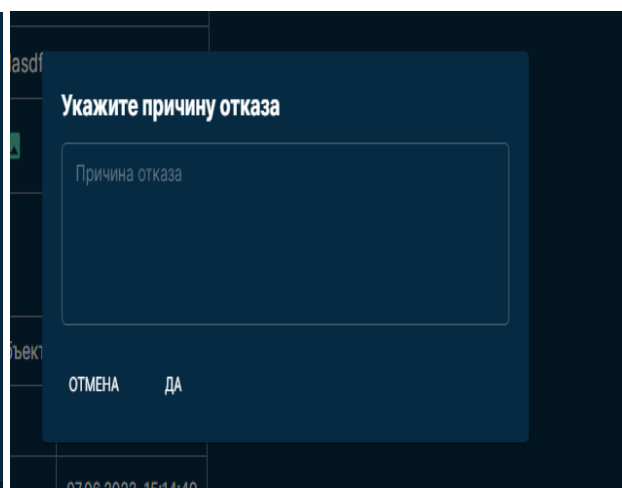
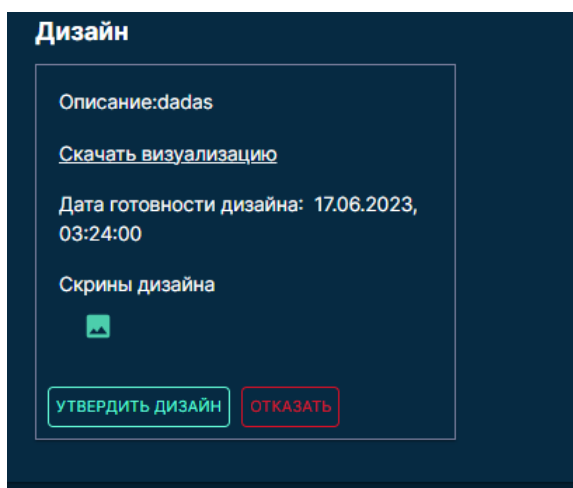


Рисунок 5.12. Операция «Утвердить дизайн» Рисунок 5.13. Операция «Отказать дизайн»

5.4.13. Операция «Прикрепление акта с прейскурантом цен»

1) Условия при соблюдении которых возможно выполнение:

- Менеджер должен быть авторизован в системе и статус заказа должен быть: Дизайн утвержден.
- Акт с прейскурантом цен должен быть составлен и подготовлен для прикрепления.

2) Подготовительное действие:

- Вход в систему менеджера с использованием учетных данных.
- Получение акта с прейскурантом цен от компании.

3) Основные действия:

- Переход на страницу управления заказами для прикрепления акта с прейскурантом цен.
- Выбор соответствующего заказа, к которому будет прикреплен акт с прейскурантом цен.
- Загрузка акта с прейскурантом цен в систему (pdf, docx).
- Проверка правильности загруженного акта и его соответствие к выбранному заказу.

4) Заключительные действия:

- Сохранение прикрепленного акта с прейскурантом цен в системе.
- Обновление информации о заказе, включая информацию о прейскуранте цен.
- Уведомление клиента о прикреплении акта с прейскурантом цен, если необходимо.
- Дальнейшая обработка заказа на основе информации, содержащейся в акте.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство менеджера.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные, предоставленные менеджеру для авторизации.
- Акт с прейскурантом цен, составленный и подготовленный для прикрепления.

Акт с прейскурантом

Прикрепите акт

ПРИКРЕПИТЕ АКТ

Документ выбран

ОТПРАВИТЬ

Рисунок 5.14. Операция «Прикрепление акта с прейскурантом цен»

5.4.14. Операция «Утверждение или отказ акта с прейскурантом»

1) Условия при соблюдении которых возможно выполнение:

- Клиент должен быть авторизован в системе и иметь соответствующие права доступа для утверждения или отказа акта с прейскурантом.
- Акт с прейскурантом должен быть предоставлен клиенту для рассмотрения.

2) Подготовительное действие:

- Вход в систему клиента с использованием учетных данных.
- Получение акта с прейскурантом от менеджера.

3) Основные действия:

- Переход на страницу управления заказами или специальную страницу для рассмотрения акта с прейскурантом.
- Ознакомление с содержимым акта (pdf, docx).
- Рассмотрение и анализ акта с прейскурантом, учитывая собственные потребности, бюджет и другие факторы.
- Принятие решения об утверждении или отказе акта с прейскурантом.

4) Заключительные действия:

- В случае утверждения акта, подтверждение данного решения в системе.
- Уведомление менеджера и других участников процесса о принятом решении.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство клиента.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные, предоставленные клиенту для авторизации.

- Акт с прејскурантом, предоставленный для рассмотрения и принятия решения.

Акт с прејскурантом	
Скачать акт	Ссылка
Дата	24.03.2023, 17:01:32
Примите решение по прејскуранту	<input type="button" value="ПРИНЯТЬ"/> <input type="button" value="ОТКЛОНИТЬ"/>

Рисунок 5.15. Операция «Утверждение или отказ акта с прејскурантом»

5.4.15. Операция «Комментирование акта с прејскурантом»

1) Условия при соблюдении которых возможно выполнение:

- Пользователь должен быть авторизован в системе и иметь соответствующие права доступа для оставления комментария.
- Акт с прејскурантом должен быть создан и доступен для комментирования.

2) Подготовительное действие:

Вход в систему соответствующего пользователя.

3) Основные действия:

- Переход на страницу заказа в раздел с актом с прејскурантом, подлежащего комментированию.
- Написание комментария в текстовом поле, предоставленном системой.

4) Заключительные действия:

- Сохранение комментария в системе.
- Отображение комментария в списке комментариев, связанных с актом.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство пользователя.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные, предоставленные пользователю для авторизации.
- Текстовое поле для ввода комментария.

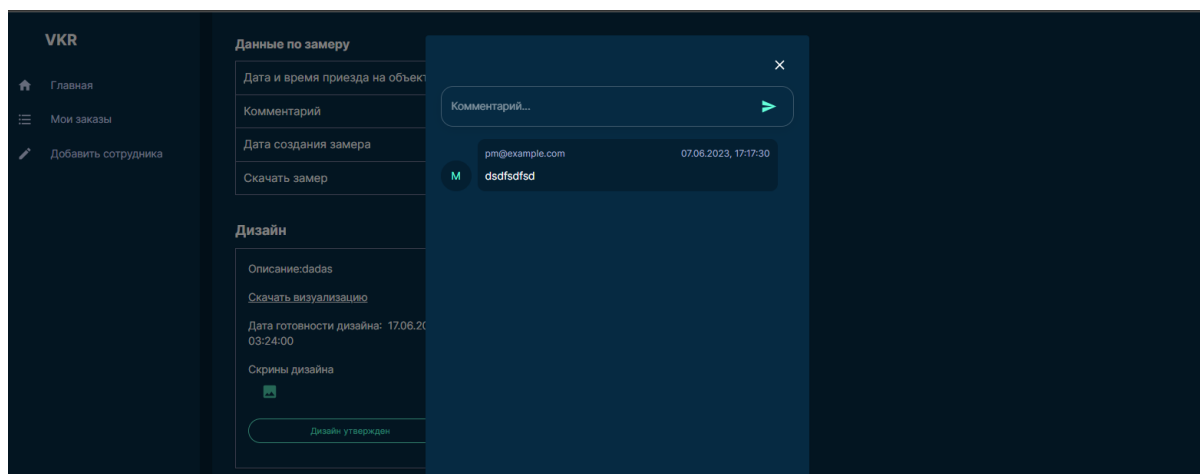


Рисунок 5.16. Операция «Комментирование акта с преискурантом»

5.4.16. Операция «Прикрепление договора»

1) Условия при соблюдении которых возможно выполнение:

- Менеджер должен быть авторизован в системе и иметь соответствующие права доступа для прикрепления договора.
- Договор должен быть подготовлен в формате PDF или DOCX и доступен для прикрепления.

2) Подготовительное действие:

Вход в систему менеджера с использованием учетных данных.

3) Основные действия:

- Переход на страницу управления заказами.
- Выбор соответствующего заказа, к которому будет прикреплен договор.
- Загрузка договора в систему. Прикрепление файла в формате PDF или DOCX.
- Проверка правильности загруженного договора и его соответствие к выбранному заказу.

4) Заключительные действия:

- Сохранение прикрепленного договора в системе.
- Обновление информации о заказе, включая информацию о договоре.
- Уведомление клиента о прикреплении договора.
- Дальнейшая обработка заказа на основе информации, содержащейся в договоре.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство менеджера.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные, предоставленные менеджеру для авторизации.
- Договор, подготовленный и доступный для прикрепления в формате PDF или DOCX.

Договор

Прикрепите договор

ПРИКРЕПИТЕ ДОГОВОР

ОТПРАВИТЬ

Рисунок 5.17. Операция «Прикрепление договора»

5.4.17. Операция «Прикрепление чека»

1) Условия при соблюдении которых возможно выполнение:

Клиент должен быть авторизован в системе.

2) Подготовительное действие:

Вход в систему клиентом.

3) Основные действия:

- Переход на страницу заказа, к которому необходимо прикрепить чек.
- Нажатие на кнопку "Прикрепить чек."
- Выбор способа прикрепления чека, загрузка файла с компьютера.
- Проверка правильности прикрепленного чека.

4) Заключительные действия:

- Сохранение прикрепленного чека в системе и его привязка к соответствующему договору.
- Отображение информации о прикрепленном чеке на странице договора.
- Уведомление менеджера и других ответственных лиц о прикреплении чека.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство клиента.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.

- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные, предоставленные клиенту для авторизации.
- Возможность загрузки файла с компьютера или ввода ссылки на чек.

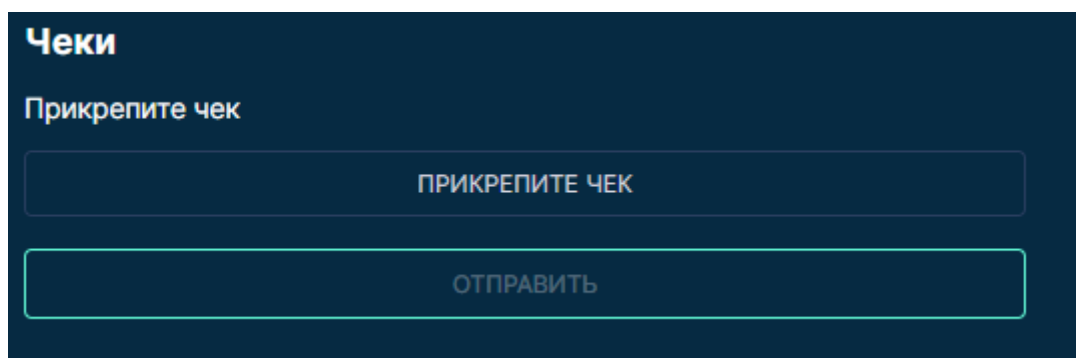


Рисунок 5.18. Операция «Прикрепление чека»

5.4.18. Операция «Принять или отклонить чек об оплате»

1) Условия при соблюдении которых возможно выполнение:

- Пользователь должен иметь соответствующие права доступа для принятия или отклонения чека об оплате.
- Чек об оплате должен быть прикреплен к соответствующему заказу или договору.

2) Подготовительное действие:

Вход в систему с соответствующими учетными данными.

3) Основные действия:

- Переход на страницу заказа, к которому прикреплен чек об оплате.
- Отображение фотографии чека.
- Рассмотрение и оценка чека об оплате в соответствии с требованиями и политикой компании.
- Принятие чека об оплате путем подтверждения его достоверности и соответствия.
- Отклонение чека об оплате, если имеются основания для отказа (например, недостаточная сумма, некорректные данные и т.д.).

4) Заключительные действия:

- Сохранение принятого или отклоненного статуса чека об оплате в системе.
- Уведомление клиента о принятии или отклонении чека об оплате, если предусмотрено системой.

- Обновление статуса заказа или договора на основе принятого или отклоненного чека об оплате.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство пользователя.
- Веб-браузер или приложение для взаимодействия с интерфейсом системы.
- Интернет-соединение для отправки запросов и получения ответов от сервера.
- Учетные данные, предоставленные пользователю для авторизации.
- Информация о чеке об оплате, включая его содержание и детали платежа.

Чеки					
Чек 1		Чек 2		Чек 3	
Скачать чек	Ссылка	Скачать чек	Ссылка	Скачать чек	Ссылка
Статус чека	Чек отклонен	Статус чека	Чек утвержден	Статус чека	Чек не обработан
				Примите решение по чеку	<input type="button" value="принять"/> <input type="button" value="отклонить"/>

Рисунок 5.19. Операция «Принять или отклонить чек об оплате»

5.4.19. Операция «Отправка отчета по ремонту»

1) Условия при соблюдении которых возможно выполнение:

Строитель должен быть ответственным за выполнение работ по ремонту.

2) Подготовительное действие:

Вход в систему с соответствующими учетными данными строителя.

3) Основные действия:

- Создание заголовка отчета, содержащего информацию о заказе/объекте ремонта и краткое описание отчета.
- Описывание проделанной работы, включая детали и особенности ремонта, использованные материалы и техники.
- Прикрепление фотографий, иллюстрирующих выполненную работу и состояние объекта до и после ремонта.

4) Заключительные действия:

Сохранение отчета в системе или его архивирование для последующего доступа и обработки.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство строителя.
- Веб-браузер или приложение для доступа к интерфейсу системы.
- Интернет-соединение для отправки отчета и получения подтверждения доставки.
- Учетные данные, предоставленные строителю для авторизации.
- Информация о выполненных работах, материалах и других деталях, необходимых для составления отчета.
- Фотографии, иллюстрирующие выполненную работу и состояние объекта до и после ремонта.

Отчеты по строительству

Оглавление

Введите оглавление отчета

Напишите описание отчета

Напишите описание отчета

Прикрепите фотографии для отчета

ПРИКРЕПИТЕ ФОТОГРАФИИ ДЛЯ ОТЧЕТА

ОТПРАВИТЬ








Отчет № 1	Отчет № 2	Отчет № 3
Оглавление:dsffsd	Оглавление:dsfdsf	Оглавление:fasdfasdf
Описание:fsdfsdfsdf	Описание:fsdfsdf	Описание:fasdfaf
Дата создания отчета: 26.05.2023, 14:05:34	Дата создания отчета: 26.05.2023, 14:18:21	Дата создания отчета: 07.06.2023, 21:55:06
Прикрепленные фотографии   	Прикрепленные фотографии   	Прикрепленные фотографии 

Рисунок 5.20 Операция «Отправка отчета по ремонту»

5.4.20. Операция «Прикрепление акта о выполненных работ»

1) Условия при соблюдении которых возможно выполнение:

- Менеджер должен быть ответственным за управление процессом ремонта.
- Работы по ремонту должны быть завершены.

2) Подготовительное действие:

Вход в систему с соответствующими учетными данными менеджера.

3) Основные действия:

- Создание акта о выполненных работ, включающего информацию о клиенте, объекте ремонта и перечне выполненных работ.
- Формирование акта в одном из форматов: PDF или DOCX.

4) Заключительные действия:

- Прикрепление созданного акта в выбранном формате (PDF или DOCX) к соответствующему заказу или объекту ремонта в системе.
- Сохранение акта в системе или его архивирование для последующего доступа и обработки.

5) Ресурсы, расходуемые на операцию:

- Компьютер, планшет или мобильное устройство менеджера.
- Веб-браузер или приложение для доступа к интерфейсу системы.
- Интернет-соединение для загрузки и прикрепления акта.
- Учетные данные, предоставленные менеджеру для авторизации.
- Шаблон акта в формате PDF или DOCX.

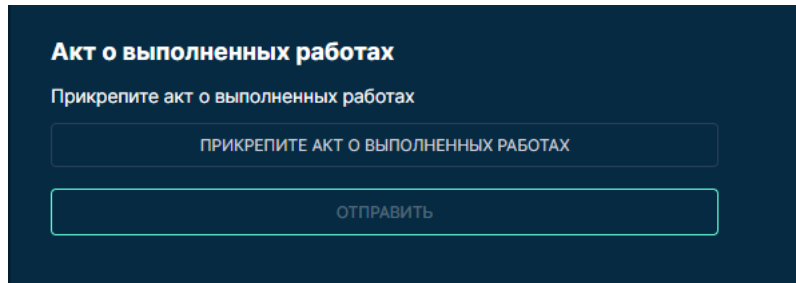


Рисунок 5.22. Операция «Прикрепление акта о выполненных работ»

5.5. Аварийные ситуации

При отказе или сбое в работе Системы необходимо обратиться к Системному администратору

Заключение

В процессе выполнения ВКР были выполнены следующие работы:

- проведен анализ деятельности предприятия;
- определены процессы подлежащие автоматизации;
- разработаны программные средства для автоматизации бизнес-процессов компании по ремонту квартир.

Разработанные программные средства выполняют следующие функции.

- Управление проектами. Система для автоматизации бизнес-процессов позволяет эффективно планировать и контролировать выполнение проектов ремонта квартир. Она предоставляет инструменты для создания заказа, назначения ответственных лиц, отслеживания статуса выполнения заказа, управления ресурсами и контроля сроков.
- Оптимизация бизнес-процессов. Система помогает оптимизировать бизнес-процессы, связанные с ремонтом квартир, путем автоматизации повторяющихся задач. Это включает прикрепление договоров и актов, отслеживание заказов и их обработка.
- Управление клиентскими отношениями. Система обеспечивает управление данными о клиентах, заказах и контактах. Она позволяет хранить и обрабатывать информацию о клиентах, отслеживать историю взаимодействия, управлять запросами и обращениями, а также повышать уровень обслуживания клиентов.
- Улучшение коммуникации и сотрудничества. Система способствует улучшению коммуникации и сотрудничества внутри компании. Она позволяет сотрудникам взаимодействовать, обмениваться информацией, делиться документацией.

В результате разработанной системы компания по ремонту квартиры может повысить производительность, повысить уровень обслуживания клиентов и свои предпочтения в бизнес-процессах. Реализованная система дает основание для активного развития и роста компании, преследуя конкурентное преимущество на рынке.

Список использованных источников

1. Позиционирование продукта – URL: <https://habr.com/ru/company/acronis/blog/516496/> (дата обращения: 31.03.2023)
2. Сведение о профилях пользователя – URL: <https://learn.microsoft.com/ru-ru/windows/win32/shell/about-user-profiles> (дата обращения: 31.03.2023)
3. Модели качества программного обеспечения – URL: https://www.kaznu.kz/content/files/news/folder22810/%D0%9B%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%20-06_2020.pdf (дата обращения: 31.03.2023)
4. Системы качества менеджмента – URL: <https://kurskmed.com/upload/files/GOST%20R%20IS%D0%9E%209001-2015.pdf> (дата обращения: 31.03.2023)
5. Software life cycle progresses – URL: http://sewiki.ru/ISO/IEC_12207 (дата обращения: 31.03.2023)
6. Оценка процессов – URL: <https://docs.cntd.ru/document/1200076921> (дата обращения: 01.04.2023)
7. Системы менеджмента информационной безопасности – URL: [https://pgm-online.com/assets/files/pubs/translations/std/iso-mek-27001-2013\(rus\).pdf](https://pgm-online.com/assets/files/pubs/translations/std/iso-mek-27001-2013(rus).pdf) (дата обращения: 01.04.2023)
8. Десять самых критических угроз безопасности веб-приложений – URL: https://wiki.owasp.org/images/9/96/OWASP_Top_10-2017-ru.pdf (дата обращения: 01.04.2023)
9. С.А. Орлов «Технологии разработки программного обеспечения. Разработка сложных программных систем» - СПб: Питер, 2002. – 464 с.: ил. (ISBN 5-94723-145-X);
10. Построение диаграммы классов – URL: https://flexberry.github.io/ru/gpg_class-diagram.html (дата обращения: 25.04.2023)
11. Простое руководство по диаграммам компонентов – URL: <https://creately.com/blog/ru/uncategorized-ru/%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5-%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5-%D0%BF%D0%BE-%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE%D0%B9-%D0%B4%D0%B8%D0%B0%D0%B3/> (дата обращения: 25.04.2023)

12. Диаграмма последовательности – URL: http://it-gost.ru/articles/view_articles/94 (дата обращения: 25.04.2023)

Приложение 1. Глоссарий

Отчетность - информация о выполненных работах, затратах, прогрессе проектов и других признаках деятельности компании по ремонту квартир.

Фреймворк - структура или набор инструментов и библиотек, создающих основу для разработки ПО.

Объект - конкретная квартира, предназначенная для проведения ремонтных работ.

Планирование тестирования - документ, описывающий потребности и подход к удовлетворению потребностей.

База данных - структурированная коллекция данных, организованная и хранящаяся в электронном формате.

Фронтенд (Frontend) — часть программного обеспечения, отвечающая за взаимодействие взаимодействия и взаимодействия пользователей с системой.

Бэкенд (Backend) — часть программного обеспечения, которая отвечает за обработку и управление данными, бизнес-логику и взаимодействие с базой данных.

Авторизация - процесс проверки и подтверждения прав доступа пользователя к поиску ресурсов или функциональности системы.

Формат файла - структура и спецификация, требующая организации данных в файле и способе их хранения. Формат файла определяет тип данных, которые могут быть сохранены в файле, а также правила и предложения для их представления и сочетания.

Акт о выполненных работах - документ, который содержит факт выполнения определенных работ или услуг в рамках проекта ремонта квартиры. Акт о выполненных работах содержит информацию о выполненных работах, описание выполненной работы, затратах ресурсов, сроках выполнения и других подробностях. Он служит опорой для подтверждения и фиксации факта выполнения работ, а также может быть зарегистрирован для оплаты труда, контроля качества и учета в проектной документации.

Договор - юридический документ, заключенный между двумя или более общими, который устанавливает условия и права, связанные с выполнением работ или осуществлением услуг.

Приложение 2. Листинг

Мидлвейр для защищенных роутов

```
const AuthGuard = ({
  children,
  roles,
}): {
  children: React.ReactNode
  roles: string[]
} => {
  const token: string | null = window.localStorage.getItem("token")
  const role = useAppSelector(selectUserRole) // нынешняя роль юзера
  const userHasRequiredRole = roles.includes(role) // проверка юзера на данную роль для доступа
  if (!token && !role) return <ErrorPage type={404} />
  if (token && role && !userHasRequiredRole) return <ErrorPage type={403} />
  else return children as JSX.Element
}
export default AuthGuard
```

Роутинг приложения

```
const MyRoutes = () => {
  const myRouter = useRoutes([
    AuthRouter,
    {
      path: "",
      element: <Layout />,
      children: [
        { path: "", element: <Dashboard /> },
        CreateOrderRoute,
        EditProfileRoute,
        MyOrdersRoute,
        CreateEmployerRoute,
      ],
    },
    { path: "*", element: <ErrorPage type={404} /> },
  ])
  return myRouter
}
export default MyRoutes
```

Переиспользуемая пагинационная панель

```
const CustomPagination = ({
  handleChangePage,
  defaultPage,
  currentPage,
  pagesCount,
}): Props => {
  const [search, setSearch] = useState<string>(currentPage.toString())
  const isMobile = useCheckMobileScreen()
  const boundaryCount = isMobile ? 3 : 6
```

```

const handleSubmit = (e: MouseEvent<HTMLInputElement>) => {
  if (parseInt(search) > pageCount || parseInt(search) < 1) {
    return
  }
  handleChangePage(e, parseInt(search))
}
useEffect(() => {
  setSearch(currentPage.toString())
  window.scrollTo({ top: 0 })
}, [currentPage])
if (pageCount <= 1) return null
return (
  <Box
    sx={{
      pb: "70px",
    }}
  >
    <Stack direction="row" spacing={2}>
      <Pagination
        count={pageCount}
        page={currentPage}
        onChange={handleChangePage}
        boundaryCount={boundaryCount}
        defaultPage={defaultPage}
        variant="outlined"
        shape="rounded"
        sx={{
          "& li button": {
            borderColor: "#64ffda",
          },
          "& li button.Mui-selected": {
            backgroundColor: "rgb(100 255 218 / 10%)",
          },
        }}
      />
      <div>
        <FormControl
          sx={{
            width: "60px",
          }}
          error={parseInt(search) > pageCount || parseInt(search) < 1}
          variant="outlined"
        >
          <OutlinedInput
            value={search}
            onChange={(e: ChangeEvent<HTMLInputElement>) =>
              setSearch(e.target.value)
            }
            onKeyUp={event => {
              if (event.key === "Enter") handleSubmit(event as any)
            }}
            type="number"
            inputProps={{
              style: {

```

```

        padding: "6px",
        height: "100%",
      },
    ]}
  endAdornment={
    <i
      onClick={handleSubmit}
      className="fa-solid fa-magnifying-glass touchable_icon"
    />
  }
/>
</FormControl>
</div>
</Stack>
</Box>
)
}
export default CustomPagination

```

Переиспользуемый компонент для промпта с комментарием

```

const PromptWithCommentModal = ({
  open,
  handleClose,
  text,
  placeholder,
  agreeCallback,
}): {
  open: boolean
  handleClose: () => void
  text: string
  placeholder: string
  agreeCallback: (text: string) => void
} => {
  const [message, setMessage] = useState("")
  return (
    <Modal sx={{ overflowY: "scroll" }} open={open} onClose={handleClose}>
      <Box
        sx={{
          maxWidth: "500px",
          margin: "200px auto 10px auto",
          background: tokensDark.primary[500],
          padding: "20px",
          "@media(max-width:640px)": {
            maxWidth: "95%",
          },
        }}
        component={Paper}
      >
        <Typography sx={{ fontSize: "18px", fontWeight: 700 }}>
          {text}
        </Typography>
        <MyTextArea
          value={message}

```

```

placeholder={placeholder}
name="message"
onBlur={() => {}}
labelName=""
onChange={(e: any) => setMessage(e.target.value)}
/>
<Stack direction="row" spacing={2} sx={{ mt: "20px" }}>
  <Button sx={{ color: "#fff" }} onClick={handleClose}>
    Отмена
  </Button>
  <Button
    sx={{ color: "#fff" }}
    onClick={() => {
      agreeCallback(message)
      handleClose()
    }}
  >
    Да
  </Button>
</Stack>
</Box>
</Modal>
)
}

```

export default PromptWithCommentModal

УТИЛИТЫ ДЛЯ ВАЛИДАЦИИ ФОРМ

```

class FormikCustomValidator {
  phoneWithout996Checker(value: string | undefined) {
    if (!value) {
      return false
    }
    if (value?.length === 10 && operators.includes(value.slice(1, 3))) return true
    return false
  }
  phoneChecker(value: string | undefined) {
    if (!value) {
      return false
    }
    if (value?.length === 13 && operators.includes(value.slice(4, 6)) && Number.isInteger(Number(value))) return
    true
    return false
  }
  innChecker(value: string | undefined) {
    if (!value) return false
    if (value?.length === 14 && Number(value[0]) < 3 && Number(value[0]) > 0 && Number(value.slice(1, 3)) <
    32 && Number(value.slice(2, 3)) < 13 && Number(
      value.slice(5, 9)) > 1920 && Number(value.slice(5, 9)) > 1920 && Number(value.slice(5, 9)) < new
    Date().getFullYear() - 15) return true
    return false
  }
  checkSizeAttachedFiles(value: FileList, size: number) {

```

```

    if (!value) {
      return false
    }
    const mb = 1024 * 1024
    let files: File[] = Array.from(value)
    let fileSize = 0
    for (let i = 0; i < files.length; i++) {
      fileSize += files![i].size
    }
    if (fileSize <= size * mb) {
      return true
    }
    return false
  }
}
checkSizeAttachedFile(value: File, size: number) {
  if (!value) {
    return false
  }
  const mb = 1024 * 1024
  if (value.size <= size * mb) {
    return true
  }
  return false
}
}
export const formikCustomValidator = new FormikCustomValidator()

```

Утилита для форматирования даты

```

const getFormattedDate = (initDate: string): string => {
  const formattedDate = addHours(
    new Date(initDate),
    getTimeZone()
  ).toLocaleString()
  return formattedDate
}
export default getFormattedDate

```

Определение типа пользователя

```

export function getRole(role: Roles) {
  let result = ""
  switch (role) {
    case Roles.BUILDER:
      result = "Строитель"
      break;
    case Roles.DESIGNER:
      result = "Дизайнер"
      break;
    case Roles.MEASURE:
      result = "Замерщик"
      break;
    case Roles.SUPERADMIN:
      result = "Суперадмин"
      break;
  }
}

```

```

    case Roles.ПМ:
      result = "Менеджер"
      break;
    case Roles.CLIENT:
      result = "Клиент"
      break;
    default:
      result = ""
      break;
  }
  return result
}

```

Утилита для преобразовывания из File в base64 строку

```

export const toBase64 = (file: File): Promise<string> => {
  return new Promise<string>((resolve, reject) => {
    const fileReader = new FileReader()
    fileReader.readAsDataURL(file)

    fileReader.onload = () => {
      resolve(fileReader.result?.toString() || "")
    }
    fileReader.onerror = error => {
      reject(error)
    }
  })
}

```

Статусы заказа

```

export enum RepairStatus {
  NEW = 'new',
  APPROVED = "approved",
  DENIED = "denied",
  MEASURE_TIME = "measure_time",
  MEASURE_ATTACHED = "measure_attached",
  DESIGN_TIME = "design_time",
  DESIGN_ATTACHED = "design_attached",
  DESIGN_APPROVED = "design_approved",
  DESIGN_DENIED = "design_denied",
  PREWORK_ATTACHED = 'prework_attached',
  PREWORK_APPROVED = 'prework_approved',
  PREWORK_DENIED = 'prework_denied',
  CONTRACT_ATTACHED = "contract_attached",
  CHECK_ATTACHED = "check_attached",
  CHECK_DECLINED = "check_declined",
  CHECK_APPROVED = "check_approved",
  FINISH_DOC_ATTACHED = "finish_doc_attached",
  STAGE_REPORT_ATTACHED="stage_report_attached"
}

```

Ссылки на страницы

```

const shared = [{ text: "Главная", href: "/", roles: [Roles.CLIENT, Roles.PM, Roles.BUILDER, Roles.MEASURE, Roles.BUILDER, Roles.DESIGNER], icon: HomeIcon }]
const client = [
  { text: "Сделать заказ", href: `create_order`, roles: [Roles.CLIENT], icon: CreateIcon },
  { text: "Мои заказы", href: `orders`, roles: [Roles.CLIENT, Roles.PM, Roles.BUILDER, Roles.MEASURE, Roles.BUILDER, Roles.DESIGNER], icon: FormatListBulletedIcon },
]
const pm = [
  { text: "Добавить сотрудника", href: `add_employees`, roles: [Roles.PM], icon: CreateIcon },
]
const Links = [
  ...shared,
  ...client,
  ...pm,
]
export default Links

```

Ограничения доступа к определенной функциональности

```

const permissions = {
  CAN_DELETE: data.status === RepairStatus.NEW && role === Roles.CLIENT,
  CAN_HANDLE: role === Roles.PM && data.status === RepairStatus.NEW,
  CAN_SET_TIME_MEASURE: role === Roles.MEASURE && data.status === RepairStatus.APPROVED,
  CAN_ATTACH_MEASURE: role === Roles.MEASURE &&
    data.status === RepairStatus.MEASURE_TIME,
  CAN_NOT_SEE: me.type === Roles.CLIENT && me.id !== data.client?.id,
  CAN_SET_TIME_DESIGN: role === Roles.DESIGNER && [RepairStatus.MEASURE_ATTACHED,
RepairStatus.DESIGN_TIME, RepairStatus.DESIGN_DENIED].includes(data.status as RepairStatus) &&
(data.designer.id === me.id && data.design.some((design) => design.is_approved === false) || data.design.length <
1),
  CAN_SET_DESIGN: role === Roles.DESIGNER && data.status === RepairStatus.DESIGN_TIME &&
data.designer.id === me.id,
  CAN_HANDLE_DESIGN: role === Roles.CLIENT && data.status === RepairStatus.DESIGN_ATTACHED
&& data.client.id === me.id,
  CAN_WATCH_PREWORK: [RepairStatus.DESIGN_APPROVED, RepairStatus.PREWORK_APPROVED,
RepairStatus.PREWORK_ATTACHED,
RepairStatus.PREWORK_DENIED, RepairStatus.CONTRACT_ATTACHED, RepairStatus.FINISH_DOC_ATTAC
HED, RepairStatus.CHECK_APPROVED].includes(data.status as RepairStatus),
  CAN_ATTACH_PREWORK: me.id === data.manager?.id && data.status ===
RepairStatus.DESIGN_APPROVED && !data.pre_work_doc?.length,
  CAN_REATTACH_PREWORK: me.id === data.manager?.id && data.status ===
RepairStatus.PREWORK_DENIED,
  CAN_HANDLE_PREWORK: role === Roles.CLIENT && data.status ===
RepairStatus.PREWORK_ATTACHED && data.client.id === me.id,
  CAN_CREATE_CONTRACT: me.id === data.manager?.id && data.status ===
RepairStatus.PREWORK_APPROVED && !data.doc_text,
  CAN_CREATE_CHECK: me.id === data.client?.id && [RepairStatus.CONTRACT_ATTACHED,
RepairStatus.CHECK_DECLINED, RepairStatus.CHECK_APPROVED].includes(data.status as RepairStatus),
  CAN_WATCH_CHECK: [RepairStatus.CHECK_ATTACHED, RepairStatus.CHECK_DECLINED,
RepairStatus.CHECK_APPROVED, RepairStatus.CONTRACT_ATTACHED].includes(data.status as
RepairStatus),
  CAN_HANDLE_CHECK: role === Roles.PM && data.status === RepairStatus.CHECK_ATTACHED &&
data.manager.id === me.id,

```



```

        CAN_CREATE_STAGE: data.builders?.find((builder) => builder.id == me.id) &&
[RepairStatus.CONTRACT_ATTACHED, RepairStatus.CHECK_APPROVED,
RepairStatus.CHECK_ATTACHED,
RepairStatus.CHECK_DECLINED, RepairStatus.STAGE_REPORT_ATTACHED].includes(data.status as
RepairStatus),
        CAN_WATCH_STAGE:
[RepairStatus.CONTRACT_ATTACHED, RepairStatus.STAGE_REPORT_ATTACHED,
RepairStatus.CHECK_APPROVED, RepairStatus.CHECK_ATTACHED,
RepairStatus.CHECK_DECLINED].includes(data.status as RepairStatus),
        CAN_CREATE_FINISH_DOC: me.id === data.manager?.id &&
[RepairStatus.CHECK_APPROVED, RepairStatus.STAGE_REPORT_ATTACHED,
RepairStatus.CHECK_ATTACHED, RepairStatus.CHECK_DECLINED].includes(data.status as RepairStatus),
        CAN_WATCH_FINISH_DOC:
[RepairStatus.CHECK_APPROVED, RepairStatus.STAGE_REPORT_ATTACHED,
RepairStatus.CHECK_ATTACHED, RepairStatus.CHECK_DECLINED].includes(data.status as RepairStatus),
    }

```

Утилита для работы с облачным хранилищем для медиа файлов
cloudinary.config(

```

    cloud_name=settings.CLOUD_NAME,
    api_key=settings.CLOUDINARY_API_KEY,
    api_secret=settings.CLOUDINARY_API_SECRET,
)

```

Helper function to delete file from cloudinary

```

def delete_file(public_id):
    response = cloudinary.api.delete_resources(public_id)
    print("Pre work doc file is deleted", response)

```

Helper function to upload file to cloudinary

```

def upload_file_docx(doc_file):
    response = cloudinary.uploader.upload(doc_file.file, resource_type="raw")
    return response["public_id"]

```

```

async def upload_file(file: UploadFile = File(...)):
    response = upload(file.file, resource_type='raw')
    file_url = response.get('secure_url')
    return file_url

```

async def upload_base64(data: str):

```

    response = upload(data)
    file_url = response.get('secure_url')
    return file_url

```

async def upload_images(images: list[dict]):

```

    async def upload_single_image(image: dict):
        response = await asyncio.to_thread(upload, image.get('image'))
        file_url = response.get('secure_url')
        image['image'] = file_url
        return image

```

try:

```

    urls = await asyncio.gather(*[upload_single_image(image) for image in images])

```

except Exception as e:

```

    raise HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR, detail=str(e))
return urls

```

```

async def delete_files_after_order_delete(public_ids):
    async def delete_files(public_id):
        print(public_id)
        response = await asyncio.to_thread(cloudinary.api.delete_resources, public_id)
        print(f"File is deleted", response)
        return response
    try:
        responses = await asyncio.gather(
            *[delete_files(public_id) for public_id in public_ids])
    except Exception as e:
        raise HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR, detail=str(e))
    return responses

```

Модуль авторизации

```

@auth_router.post("/register", status_code=status.HTTP_201_CREATED,
response_model=schemas.UserResponse)
async def create_user(
    payload: schemas.RegisterUserSchema, session: AsyncSession = Depends(get_async_session)
):
    user = await User.get_all(session, email=payload.email)
    if user:
        raise HTTPException(
            status_code=status.HTTP_409_CONFLICT, detail="Account already exist"
        )
    payload.password = utils.hash_password(payload.password)
    new_user = User(**payload.dict())
    await new_user.save(session)
    return new_user

```

```

@auth_router.post("/login", response_model=schemas.UserResponseWithToken)
async def login(
    payload: schemas.LoginUserSchema,
    response: Response,
    authorize: AuthJWT = Depends(),
    session=Depends(get_async_session),
):
    user = await User.get(session, email=payload.email)
    if not utils.verify_password(payload.password, user.password):
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Incorrect Email or Password",
        )
    user.access_token = authorize.create_access_token(
        subject=str(user.id), expires_time=timedelta(minutes=ACCESS_TOKEN_EXPIRES_IN)
    )
    user.refresh_token = authorize.create_refresh_token(
        subject=str(user.id), expires_time=timedelta(minutes=REFRESH_TOKEN_EXPIRES_IN)
    )
    access_keys = redis_conn.keys(f"access_token:{user.id}")
    refresh_keys = redis_conn.keys(f"refresh_token:{user.id}")
    for key in access_keys + refresh_keys:
        redis_conn.delete(key)

```

```

        redis_conn.setex(f"access_token:{user.id}", timedelta(minutes=ACCESS_TOKEN_EXPIRES_IN),
user.access_token)
        redis_conn.setex(f"refresh_token:{user.id}", timedelta(minutes=REFRESH_TOKEN_EXPIRES_IN),
user.refresh_token)
    response.set_cookie(
        "access_token",
        user.access_token,
        ACCESS_TOKEN_EXPIRES_IN * 60,
        ACCESS_TOKEN_EXPIRES_IN * 60,
        "/",
        None,
        True,
        True,
        "none",
    )
    response.set_cookie(
        "refresh_token",
        user.refresh_token,
        REFRESH_TOKEN_EXPIRES_IN * 60,
        REFRESH_TOKEN_EXPIRES_IN * 60,
        "/",
        None,
        True,
        True,
        "none",
    )
    response.set_cookie(
        "logged_in",
        "True",
        ACCESS_TOKEN_EXPIRES_IN * 60,
        ACCESS_TOKEN_EXPIRES_IN * 60,
        "/",
        None,
        True,
        False,
        "none",
    )
    return user

```

```

@auth_router.get("/refresh")
async def refresh_token(
    response: Response,
    authorize: AuthJWT = Depends(),
    session=Depends(get_async_session),
):
    try:
        authorize.jwt_refresh_token_required()
        user_id = authorize.get_jwt_subject()
        if not user_id:
            raise HTTPException(
                status_code=status.HTTP_401_UNAUTHORIZED,
                detail="Could not refresh access token",
            )
    except:

```

```

user = await User.get(session, id=int(user_id))
if not user:
    raise HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="The user belonging to this token no longer exist",
    )
refresh_token_key = f"refresh_token:{user_id}"
print(authorize.get_jti(redis_conn.get(refresh_token_key)), 'token_from_redis')
print(authorize.get_raw_jwt()['jti'], 'user_token')
print(authorize.get_jti(redis_conn.get(refresh_token_key)) == authorize.get_raw_jwt()['jti'], 'is_equal')
print(authorize.get_raw_jwt())
if authorize.get_jti(redis_conn.get(refresh_token_key)) != authorize.get_raw_jwt()['jti']:
    raise HTTPException(status_code=401, detail='Token expired or invalid')
access_token = authorize.create_access_token(
    subject=str(user.id),
    expires_time=timedelta(minutes=ACCESS_TOKEN_EXPIRES_IN),
)
access_keys = redis_conn.keys(f"access_token:{user.id}")
for key in access_keys:
    redis_conn.delete(key)
redis_conn.setex(f"access_token:{user.id}", timedelta(minutes=ACCESS_TOKEN_EXPIRES_IN),
access_token)
except Exception as e:
    error = e.__class__.__name__
    if error == "MissingTokenError":
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Please provide refresh token",
        )
    raise HTTPException(
        status_code=status.HTTP_400_BAD_REQUEST, detail=error)

response.set_cookie(
    "access_token",
    access_token,
    ACCESS_TOKEN_EXPIRES_IN * 60,
    ACCESS_TOKEN_EXPIRES_IN * 60,
    "/",
    None,
    True,
    True,
    "none",
)
response.set_cookie(
    "logged_in",
    "True",
    ACCESS_TOKEN_EXPIRES_IN * 60,
    ACCESS_TOKEN_EXPIRES_IN * 60,
    "/",
    None,
    True,
    True,
    "none",
)

```

```
return {"access_token": access_token }
```

```
@auth_router.get("/logout", status_code=status.HTTP_200_OK)
def logout(
    response: Response,
    authorize: AuthJWT = Depends(),
    user_id: str = Depends(oauth2.require_user),
):
    authorize.unset_jwt_cookies()
    response.set_cookie(
        "logged_in",
        "",
        -1,
        -1,
        "/",
        None,
        True,
        True,
        "none",
    )

    return {"status": "success" }
```

Модели связанные с заказом

```
class OrderBase(BaseModel):
    address: str
    series: Series
    room_amount: int
    square: float

    class Config:
        orm_mode = True

class OrderWithIdSchema(OrderBase):
    id: int

class OrderCreateSchema(OrderBase):
    order_room: list[OrderRoomCreate]

class FinishDocSchema(BaseModel):
    id: int
    file: str | None = None
    created_at: datetime | None = None

    class Config:
        orm_mode = True

class OrderResponseSchema(OrderWithIdSchema):
    reason_of_deny: str | None = None
```

```

created_at: datetime
status: Status
order_room: list[OrderRoomResponse]
client: UserResponse | None = None
is_contract_signed: bool
meauser: UserResponse | None = None
designer: UserResponse | None = None
manager: UserResponse | None = None
builders: list[UserResponse] | None = None
doc_text: str | None = None
repair_type: RepairTypeEnum | None = None
design: list[DesignResponse] | None = None
measurement: list[MeasurementResponseSchema] | None = None
pre_work_doc: list[PreWorkDocResponseSchema] | None = None
order_check: list[OrderCheckResponse] | None = None
stage: list[StageResponse] | None = None
finish_doc: list[FinishDocSchema] | None = None

```

```

class OrderResponseListSchema(OrderWithIdSchema):

```

```

    created_at: datetime
    status: Status
    client: UserResponse | None = None

```

```

class OrderRetrieveSchema(BaseModel):

```

```

    id: int
    client: UserResponse | None = None
    meauser: UserResponse | None = None
    designer: UserResponse | None = None
    manager: UserResponse | None = None
    builders: list[UserResponse] | None = None
    reason_of_deny: str | None = None
    address: str | None = None
    series: Series | None = None
    order_room: list[OrderRoomResponse]
    room_amount: int
    status: Status
    is_contract_signed: bool
    created_at: datetime
    doc_text: str | None = None
    square: float

```

```

class Config:

```

```

    orm_mode = True

```

```

class OrderFilterSchema(BaseModel):

```

```

    id: int | None = None
    client_id: int | None = None
    meauser_id: int | None = None
    designer_id: int | None = None
    manager_id: int | None = None
    reason_of_deny: str | None = None

```

```
address: str | None = None
series: Series | None = None
room_amount: int | None = None
status: Status | None = None
is_contract_signed: bool | None = None
created_at: datetime | None = None
doc_text: str | None = None
square: float | None = None
```

```
class OrderDeclineSchema(BaseModel):
    reason_of_deny: str
```

```
class OrderAcceptSchema(BaseModel):
    id: int
    status: Status
    client: UserResponse
    meauser: UserResponse
    designer: UserResponse
    manager: UserResponse
    builders: list[UserResponse]
```

```
class Config:
    orm_mode = True
```

```
class OrderDependSchema(BaseModel):
    meauser_id: int = Field(gt=0)
    designer_id: int = Field(gt=0)
    builder_id: list[int] = Field(gt=0)
    repair_type: RepairTypeEnum
```